Faculdade de Engenharia da Universidade do Porto

Multi-Technology Bus-To-Infrastructure Wireless Networks: A Multipath Transfer Solution

Ricardo Jorge Nunes Rocha

Master in Electrical and Computers Engineering

Supervisor: Prof. Manuel Alberto Pereira Ricardo (PhD)

Co-supervisor: Helder Martins Fontes (MSc)

June, 2014

© Ricardo Jorge Nunes Rocha, 2014

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Multi-technology Bus-to-Infrastructure Wireless Networks: A Multipath Transfer Solution"

foi aprovada em provas realizadas em 22-07-2014

o júri

Presidente Professor Doutor Mário Jorge Moreira Leitão
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Manuel Tavares Vieira Cabral
Professor Associado do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Ricardo Jorge Nunes Rocha

Faculdade de Engenharia da Universidade do Porto

Abstract

In the last decade, we witnessed an increasing demand for Internet services available anytime, anywhere. People want to be able to use services such as social networks, VoIP, e-mail, among others, in the most tedious periods of their daily life, such as when travelling in public transports. Hence, it seems natural that, in recent years, there has been an increasing offer of Internet services, via Wi-Fi, to the passengers of public transports.

The project developed in this dissertation is motivated by the SITMe project, whose main goal is to provide Internet services via Wi-Fi to bus passengers, among other services, such as news and entertainment information, and geo-referenced interactive services. Each bus is able to use multiple wireless technologies, such as UMTS and Wi-Fi, to connect to the network infrastructure, but currently only supports the usage of one network interface at a time, chosen according to defined metrics.

The main goal of this dissertation was to develop a multipath transfer solution that is able to simultaneously use the available network interfaces to provide intelligent traffic distribution and load-balancing (based on defined link metrics and type of traffic), which results in a better usage of the properties of each network link, as well as higher aggregated bandwidth and fault-tolerance in the system's network.

The main goals of this project were achieved resulting in the development of a new Multipath Transfer Solution. A series of tests were performed to the developed solution, in order to assess its performance, which resulted in very favorable results. Such results led to the conclusion that the developed solution has the potential to bring a substantial improvement to the SITMe project.

Resumo

Na última década, assistimos a um crescimento na procura por serviços de Internet disponíveis a qualquer altura e em qualquer lugar. As pessoas querem ter a possibilidade de usar serviços como redes sociais, VoIP, e-mail, entre outros, nas alturas mais entediantes do seu dia-a-dia, como, por exemplo, nas viagens em transportes públicos. Portanto, é com naturalidade que, nos últimos anos, tenha havido um crescimento na oferta de serviços de Internet, através de Wi-Fi, para passageiros de transportes públicos.

O projeto desenvolvido nesta dissertação é motivado pelo projeto SITMe, cujo principal objetivo é fornecer serviços de Internet via Wi-Fi aos passageiros de autocarro, entre outros serviços, como notícias e informação de entretenimento, e serviços interativos georreferenciados. Cada autocarro é capaz de usar múltiplas tecnologias, como UMTS e Wi-Fi, para se ligar à infraestrutura de rede, mas atualmente apenas suporta o uso de uma interface de rede de cada vez, escolhida de acordo com métricas definidas.

O principal objetivo desta dissertação foi de desenvolver uma solução de transferência de dados através de múltiplos caminhos de rede, que fosse capaz de usar simultaneamente as *interfaces* de rede disponíveis, para proporcionar distribuição de tráfego e balanceamento de cargas de forma inteligente (baseado em métricas de ligação definidas e tipos de tráfego), que resultam numa melhor utilização das propriedades de cada ligação, assim como uma maior largura de banda agregada e tolerância a falhas na rede do sistema.

Os principais objetivos deste projeto foram atingidos resultando no desenvolvimento de uma nova solução chamada Multipath Transfer Solution. Uma série de testes foram realizados de forma a testar o desempenho da solução desenvolvida, sendo os resultados obtidos positivos. Tais resultados permitiram-nos concluir que a solução desenvolvida tem potencial para providenciar uma melhoria substancial ao projeto SITMe.

Acknowledgments

I would like to thank my colleagues and teachers who helped me, through all this years, to acquire most of the knowledge that was used as a basis for this dissertation.

I would especially like to express my gratitude to my supervisors Helder Martins Fontes and Prof. Manuel Alberto Pereira Ricardo for always being there to guide me and help me whenever it was needed throughout the development of this project. Their vision and experience were of great importance to the realization of this dissertation.

Finally, I would like to thank my family and friends, and especially my parents, for supporting me both emotionally and financially throughout this course. Without them, none of this would have been possible.

Table of Contents

List of Figures

List of Tables

Abbreviations

AAA	Authentication, Authorization and Accounting
DCCP	Datagram Congestion Control Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
HIP	Host Identity Protocol
ICMP	Internet Control Message Protocol
INESC Porto	Instituto de Engenharia e Sistemas de Computadores do Porto
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MPLS	Multi Protocol Label Switching
MPRTP	Multipath Real-time Transport Protocol
MPTS	Multipath Transfer Solution
NS-3	Network Simulator 3
NTP	Network Time Protocol
OS	Operating System
OSI	Open Systems Interconnection
PSTN	Public Switched Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SCTP	Stream Control Transmission Protocol
SITMe	Serviços Integrados para Transportes Metropolitanos
VIF	Virtual Network Interface
VoIP	Voice over Internet Protocol
SQSAMR	Send Queue Size Above Max Ratio
STCP	Sociedade de Transportes Colectivos do Porto
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 - Contextualization

In the last decade there has been an increasing demand for Internet services available anytime, anywhere. Services such as e-mail, social networks, VoIP, Instant Messaging, among others, are massively used nowadays, whether it is for work or leisure, so it seems natural that, in recent years, there has been an increasing offer of Internet services, via Wi-Fi, on public transports.

This dissertation is motivated by the experience of INESC Porto on the former SITMe[1] project, whose main goal was to provide Internet services via Wi-Fi to bus passengers, among other services, such as news and entertainment information, and geo-referenced interactive services. INESC Porto was a main partner involved in SITMe and the responsible for the design, implementation and maintenance of the project's communication system. The SITMe had a real testbed in operation for over a year which allowed to gather valuable experience and find limitations of the implemented solution. This dissertation is expected to produce new scientific contributions addressing some limitations of the original solution.

One of the main characteristics that distinguish SITMe from other related systems is the way the access to the infrastructure outside of the bus is made. Most systems are usually dependent on a single interface to connect to the exterior of the bus (*e.g.* UMTS connection from a mobile Internet Service Provider (ISP)), which leads to underused resources from other available networks and fault tolerance issues, since it depends on the availability of that single network. SITMe, on the other hand, is able to use multiple technologies, such as UMTS, Wi-Fi and Wi-Max to access the infrastructure, choosing the best available technology to use at the moment, according to some adjustable metrics. However, this type of multi-interface system is not ideal, because there is only one interface used at a time, which means that if there are other links available, those links are not used, leading to underutilization of the available resources.

[1] http://www.sitme.org/

A solution to this problem would be to implement a system that is able to make use of all the available links simultaneously, distributing the traffic between them and, at the same time, take advantage of the characteristics of each one of them (*e.g.* UMTS would be better for VoIP traffic, since it is more stable in terms of availability along the bus route, when compared to Wi-Fi networks).The design and implementation of the referred system is the main focus of this dissertation.

1.2 - Motivation and Goals

The main motivation for this project is the necessity to improve the SITMe's communication system by optimizing the usage of the available network resources. There are moments when more than one Bus-To-Infrastructure network interface is available, creating multiple network paths between the bus and the core Internet gateway. In the current solution only one interface is selected to exchange traffic, hence a new multi-technology system exploring the multipath characteristics of this scenario has to be developed. Although this work is focused on the SITMe project, the goal is to develop a multi-technology system generic enough so it can also be easily adapted to other heterogeneous network systems. As a result, the client and server software modules that implement the multipath transfer of the network traffic through the available links will have to be independent, but compatible with the SITMe network solution.

The main objectives of this dissertation are:

- Study the consequences of using a multipath transferring scheme with links with very different properties, such as the RTT (Round-Trip Time), packet loss and available bandwidth;
- Study the best solution to implement the multi-technology bus-to-infrastructure system (assuming that, in this case, for the sake of simplicity, it will only be used two interfaces (UMTS and Wi-Fi)) that provides intelligent traffic distribution and load-balancing (based on defined link metrics and type of traffic), which results in a better usage of the properties of each network link, as well as higher aggregated bandwidth and fault-tolerance in the system's network;
- Develop a prototype that implements the above referred multipath transfer system. This system will consist of two modules: the Client Module, operating on the buses, and the Server Module, serving as the gateway to the Internet;
- Test the solution in a simulation scenario, to validate the good operation of the developed prototype.

1.3 - Results

In this dissertation it was developed a Multipath Transfer Solution capable of providing intelligent traffic distribution and load-balancing over the available network links (based on defined link metrics and type of traffic), as well as higher aggregated bandwidth and fault-tolerance to the SITMe project's communication system. In spite of being developed to improve the SITMe project, the architecture of this solution is generic enough to be easily

adapted to other heterogeneous network systems to provide network traffic transfer that intelligently uses the multiple paths available as if they were only one link, i.e., the developed Client and Server modules can act as a proxy for end-to-end applications, providing an abstraction layer to multihomed devices, avoiding the need for changes at the application layer. A series of mechanisms were developed to provide the estimation of the link metrics used by the system, such as the packet loss and Round-Trip Time, as well as a mechanism called HELLO System that keeps the Server Module updated about the IP addresses (and other information) of the network interfaces of each Client Module, since they operate in a mobility scenario.

There were some implementation challenges, especially regarding the estimation of the available bandwidth of the network links that would be used as a metric for load-balancing. This led us to use an alternative approach, based on the occupation of the socket send queues. However, this alternative solution has some limitations that make it unsuitable for some network scenarios.

A series of tests were performed to individually evaluate the operation of the several mechanisms that constitute the Multipath Transfer Solution, as well as the overall performance of the system as a whole, using a simulation scenario that was implemented for this purpose. Despite some identified issues, the results were positive, and allowed us to conclude that the Multipath Transfer Solution is performing well and has the potential to bring a substantial improvement to the SITMe project.

1.4 - Document Structure

This document is structured in the following way: Chapter 2 presents an overview of the SITMe project, for a better contextualization and understanding of the motivation for this dissertation, as well as the study and critical analysis of the state of the art technologies which are relevant to the implementation of the developed solution; Chapter 3 presents the requirements for this project, as well as an overview of the architecture and some of the defined metrics and mechanisms that compose the proposed Multipath Transfer Solution; Chapter 4 presents the implementation of the Multipath Transfer Solution, by explaining the design decisions made and the approach used in the development of each part of the developed prototype; Chapter 5 presents a series of tests performed to individually evaluate the operation of the several mechanisms that constitute the developed solution, as well as the overall performance of the system as a whole, as well as the results obtained from each test; Finally, Chapter 6 presents the conclusions of this dissertation, as well as the known limitations of the developed solution and suggestions for addressing those limitations in future work.

Chapter 2

State of The Art

2.1 - The SITMe Project

Since this dissertation is motivated by the improvement of the SITMe project's communication system, this Section presents an overview of the SITMe project, especially focusing on the aspects of bus-to-Infrastructure (e.g. Internet Gateway access) data transferring over public links, which are the main focus of this dissertation. The objective is to produce a better understanding of the requirements and goals of the project to be developed, and the familiarization with the architecture and operation of the SITMe project's communication system.

2.1.1 - SITMe overview

The SITMe[2] project [1] [2] was developed to provide information services to bus passengers. Some of those services are Internet access by Wi-Fi, news and entertainment information, geo-referenced interactive services, inter-modality information, among others.

In order to accomplish this, it was implemented a network architecture, that is presented in Figure 2.1, where each bus has an RBridge (which operates between layer 2, of the OSI model [3], as a bridge and layer 3 as a router), that is connected to a Wi-Fi Access Point inside the bus, providing Internet access to the bus passengers. The RBridges connect through the available interfaces to the Core RBridge, which is a server that acts as a gateway to the Internet and performs some administrative functions to the system, such as DHCP, DNS and AAA.

[2] Serviços Integrados para Transportes Metropolitanos

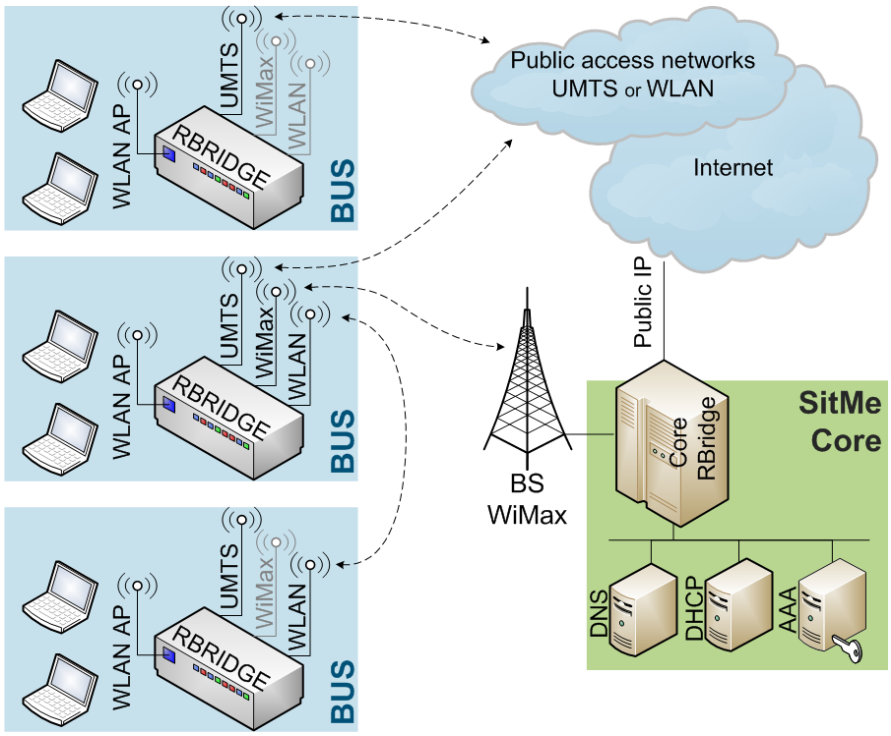

Figure 2.1 - SITMe Communication System's Overview [1]

Each RBridge uses multiple wireless network technologies to access the network infrastructure outside of the bus. Some of the currently supported technologies are UMTS, WiMax and Wi-Fi.

The RBridges are able to connect to public and private networks. The UMTS interfaces will always be connected to a public ISP network, while the WiMax and WLAN interfaces can be connected to private networks, owned and managed by SitMe, as well as public ones.

The interface switching is provided by an intelligent selection mechanism that chooses the best available interface based on economical and technical aspects. This switching is done in a way that it is not perceivable to the users inside of the bus, keeping their communication sessions unaffected.

This is achieved by the implementation of an overlay layer 2 network that makes it possible to have all the buses inside of the same virtual private network. This network is maintained by the RBridges using the Wireless Metropolitan Routing Protocol (WMRP), which is a layer 2.5 protocol, performing routing operations at the MAC level and using the MPLS header to establish virtual paths between the RBridges. Since it is not possible to transmit data encapsulated with the layer 2.5 MPLS header over public links due to restrictions imposed by ISP's network elements, UDP encapsulation, as shown in Figure 2.2, is used to exchange data between the RBridges and Core RBridge over public links (*e.g.* UMTS).

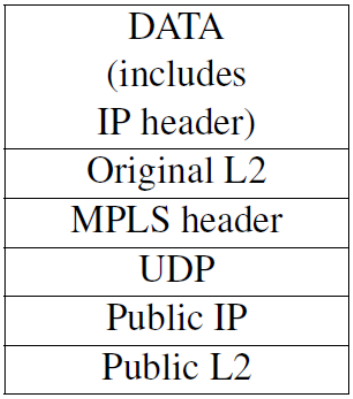

Figure 2.2 - UDP encapsulation stack in SITMe for core network links operating over public links [1]

This project had a pilot that was running for over a year, between December 2011 and February 2013. This pilot was constituted by 11 buses of the 207 line of SCTP[3] and ran with very favorable results. The RBridges used UMTS, WiMax and Wi-Fi network links as Bus-to-Infrastructure network links. The switch between technologies was seamless, allowing the bus passenger to maintain the current active Internet sessions. The testbed obtained around 12.000 individual users and, at rush hour, the system was used by approximately 5 to 6 people per bus, giving a total of roughly 50-60 simultaneous users.

One identified limitation of the SITMe project was the underutilization of network resources, because only one technology was chosen at a given moment. Also, the selection between available interfaces needs to be improved, making it possible to select the best interface available for a given type of traffic. For example, Voip traffic would usually benefit from a more stable connection with a wider coverage such as UMTS and should always be assigned to that interface, while traffic intensive flow types would benefit from the higher bitrates of more sporadic Wi-Fi links. In summary, the SITMe communication system lacked simultaneous multipath and a smarter interface selection based on the type of traffic being transported.

2.2 - Multipath Protocols

In recent years, with the increase of multihomed devices, *i.e.*, devices connected to more than one network (*e.g.* a smartphone with Wi-Fi and UMTS connections), a number of multipath protocols are being developed to provide multipath capabilities to such devices, making a better use of the available network resources.

This section presents a brief overview of some multipath solutions already developed, and a discussion of their possible application to the solution developed in this dissertation.

[3] Sociedade de Transportes Coletivos do Porto

2.2.1 - Multipath TCP

Multipath TCP [4] is a transport layer protocol that provides multipath transferring of data over more than one network interface. It is an extension of the TCP protocol and therefore it shares some of its main characteristics (connection-oriented, reliable and ordered delivery, congestion control, etc.) and is backwards-compatible with regular TCP.

Its multipath capabilities improve resource usage within the network and, thus, provide higher throughput and improved resilience to network failure.

This protocol would fit some of the requirements of this project; however, there are some issues that make it unsuitable for tunneling TCP [5] and UDP [6] traffic:

- UDP traffic is not TCP-friendly [7]. The aggravated delay created by congestion control and retransmission of lost packets would defeat the purpose of the UDP protocol, which is intended for time-sensible applications, where it is preferable to drop packets than waiting for delayed packets;
- Tunneling TCP over TCP usually degrades the goodput of end-to-end TCP flows [8], due to unnecessary retransmission of packets. For instance, if a packet loss occurs in the TCP tunnel, there will be a delay in packet transfer. Because of that delay, a retransmission timeout would occur in the end-to-end TCP connection, leading to retransmission of the lost packets by the end-to-end TCP. Since lost packets are already retransmitted by the TCP tunnel, packets retransmitted by the end-to-end TCP are of no value, which leads to decrease in the goodput of the end-to-end TCP flow.

2.2.2 - Stream Control Transfer Protocol

SCTP [9] is a transport layer protocol that was initially intended for transporting PSTN[4] signaling over IP [10] networks. It is message-oriented and provides reliable and ordered transport of messages with congestion control.

This protocol has direct support for multihoming, enabling it to establish connections between hosts over more than one network interfaces, hence increasing association survivability in the case of a network path failure, but currently is not capable of using multiple paths simultaneously. However, there has been some research in that area, aiming to provide concurrent multipath transfer to SCTP [11], using its multihoming feature to distribute data across multiple paths in a multihomed SCTP association.

This protocol is not suitable to tunnel UDP and TCP traffic, since it is a reliable transport protocol with flow and congestion control, and the delay created by those mechanisms would defeat the purpose of the UDP protocol and could degrade the goodput of the end-to-end TCP flows, as mentioned in Section 2.2.1, for Mutipath TCP.

[4] Public Switched Telephone Network

2.2.3 - Multipath RTP

MPRTP [12] is a backwards-compatible extension to the Real-time Transport Protocol (RTP) [13]. It allows spreading packets of a media stream across a number of different paths, which can be discovered and set up dynamically within an RTP session. This improves network utilization and results in an increase in reliability and throughput that can also enhance the user experience.

As regular RTP, it is a session layer protocol that runs over a transport layer protocol (usually UDP), intended for real-time applications, such as VoIP and video streaming, providing delivery of audio and video data over IP networks. Therefore, it is not suitable for this project. However, since it is a protocol that has some similarities to this project, there are some aspects that might be useful to study, such as the load balancing algorithm and control mechanisms.

2.2.4 - Datagram Congestion Control Protocol

The Datagram Congestion Control Protocol (DCCP) [14] is a transport layer protocol. It is message-oriented with no reliable in-order delivery, but provides congestion controlled flow of unreliable datagrams. It is suited for delay-sensitive applications, such as VoIP, streaming media and online video games that normally use UDP and, since UDP has no congestion control, developers usually implement their own congestion control mechanisms at the application layer, or don't implement any congestion control at all. DCCP provides an easier way to deploy this kind of applications without the risk of congestion collapse.

DCCP provides primitive support for multihoming [15] via a mechanism for transferring a connection endpoint from one address to another, making it basically only useful for simple connection migration and not for simultaneous usage of multiple interfaces.

2.2.5 - Host Identity Protocol

The Host Identity Protocol (HIP) [16] is a host identification technology that provides an alternative to the dual use of IP addresses as "locators" (routing labels) and "identifiers" (endpoint, or host, identifiers), by introducing a new namespace for the host machine known as Host Identifiers, which are based on public-key cryptography . This allows a separation of the identifier and locator roles of IP addresses, thereby enabling continuity of communications across IP address changes, since the transport layer protocols are bound to the Host Identifiers instead of IP addresses.

This protocol supports multihoming [17] , but there is little information about the usage of more than one network interface simultaneously in its specification, thus it basically works for failover scenarios, making it not suitable for this project, since the usage of multiple interfaces simultaneously is a requirement for this project.

2.3 - Available Bandwidth Estimation Tools

In a multipath transferring system it is important to have some kind of mechanism that estimates the available bandwidth of the available paths, so it can be used as a metric to

better balance the load over those paths. Since the focus of this dissertation is not to develop a new bandwidth estimation mechanism, it is important to study existing bandwidth estimation tools in order to determine the best one that can be integrated in the solution to be developed.

There are two main models used by the majority of the existing bandwidth estimation tools: the Probe Gap Model and the Probe Rate Model.

The basic idea of the Probe Gap Model is to send a pair, or train, of packets with a known spacing between them to the destination of interest and measure the changes in those spacings at the receiver to estimate the bandwidth properties of the network path [18].The Probe Rate Model is based in the concept of self-induced congestion. This means, sending a sequence of packets at a known rate, and if the available bandwidth of the path being probed is higher than the sending rate, there will be no notable variation in the probing traffic and it will match with the sender's rate. In the other hand, if the sending rate is higher than the available bandwidth of the path, there will be delays and/or packet loss in the probing traffic, which indicates path congestion. The available bandwidth is estimated by determining the turning point at which it is noted an increasing trend in the effects on the probing stream above described [19] [20].

Four existing tools were considered. Two based on the Probe Gap Model: Spruce [21] and IGI [18]; and two based on the Probe Rate Model: Pathload [19] and Yaz [20].

A performance comparison based on experiments over a large number of Internet paths [21] shows that Spruce provides more accurate estimations of the available bandwidth of a network path than Pathload and IGI. Pathload tends to overestimate the available bandwidth whereas IGI becomes insensitive when the bottleneck utilization is large.

Another comparison [20] shows that Yaz is significantly more accurate than both Spruce and Pathload, as well as faster in the estimation of the available bandwidth than those two tools. Hence, it is also a better option than IGI, since Spruce, in turn, performs better than IGI, as stated above. Yaz is also much less intrusive than Pathload in terms of load introduced to the network caused by the probing streams.

Therefore, Yaz seems to be the better choice among these four available bandwidth estimation tools. It is also an open source tool written in C/C++ programming languages, which are the same programming languages used in the development of this prototype, making it easier to adapt and integrate in the developed solution.

2.4 - Other Related Work

It is important to study some other related work to better understand the available options and get some useful ideas to apply to the solution developed in this dissertation.

2.4.1 - Video Streaming Over Multi-Radio Access Networks

Nuno Novo *et al.* [22], developed a solution to provide video streaming to mobile devices, using its multiple network interfaces in a simultaneous way, by dividing the traffic in the

server, applying a dynamic and intelligent algorithm based on the conditions of the connections established between the server and the terminal, and recovering the flow in the terminal using the multiple retrieved streams transported by the various networks.

A high level illustration of the multipath system architecture is presented in Figure 2.3

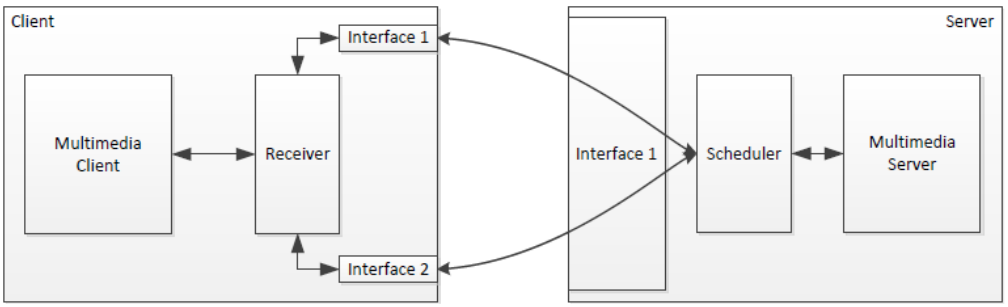

Figure 2.3 - Multipath system architecture for Video Streaming Over Multi-Radio Access Networks [22]

Although this is a solution to a different kind of system, i.e., a unidirectional multipath transfer system (server to client), where the data is completely available at the server before it starts transferring it to the client, it is an interesting approach that could, to some extent, be applied to the multipath transfer solution developed in this dissertation, such as the system architecture presented in Figure 2.3. This architecture provides an abstraction layer to the end-to-end applications, by having the Scheduler and Receiver modules act as a proxy running in the middle of the client-server communications. A similar architecture adapted to bidirectional multipath transferring was used in the solution developed in this dissertation.

2.4.2 - Transport protocols in multipath transferring schemes

In another related work, Yabandeh *et al.* [23], proposed two end-to-end streaming mechanisms to forward packets of a single flow through multiple paths; one for UDP and another for TCP connections. This was motivated by the fact that transmitting data packets over multiple paths with very different characteristics would cause the transmitted packets to experience different delays, and, as a result, to be delivered with a different order with respect to what the source has sent, leading to more demanded buffer space and extra delays for the receiver's application.

In the case of UDP, it was proposed a streaming method that schedules packets at the sender among multiple paths in a way that they arrive at the receiver in order, as illustrated in Figure 2.4, thus imposing the minimum possible delay on the receiver's application to start after the sender begins its transmission, and, at the same time, reducing the need for large buffer spaces at the receiver.

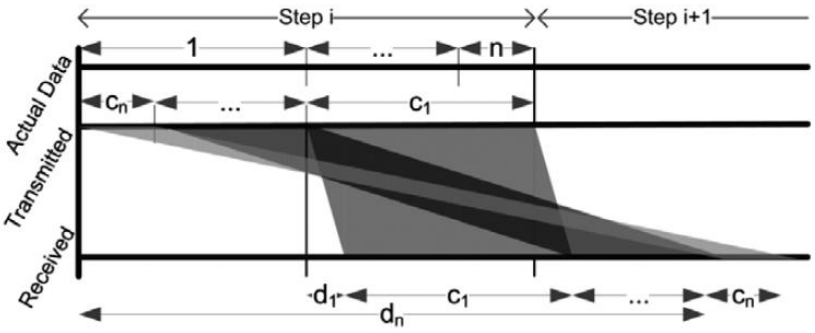

Figure 2.4 - Schematic view of the proposed scheduling for UDP connections [23]

The study of this mechanism is particularly interesting for this dissertation, since it allows to better understand some of the main issues of transferring UDP traffic over multiple paths, and provides a good solution for those issues. However, the main advantage of this solution is to be able to reduce the processing and the need for a large buffer at the receiver, which would be ideal for devices with very limited resources (*e.g.*, an old or low-end smartphone), but shouldn't be an issue to the bus and gateway-server machines used in this project. This might allow for the implementation of a simpler solution that performs the reordering of the packets at the destination.

There is also another major issue that makes this mechanism unsuitable for this project. Since the sender breaks the original order of data, the data should be completely available before the start of transmission. Consequently, this approach is not suitable for some of the services that might be used by bus passengers, like VoIP and video conferencing which consume data directly.

2.5 - Conclusions

In this chapter it was presented an overview of the SITMe project, which provided familiarization with the architecture and operation of the SITMe project's communication system, as well as allowing a better understanding of the requirements and goals of the project to be developed in this dissertation, since it is motivated by the need to improve some of the limitations of the SITMe project.

It was also studied several multipath protocols and other related work, and discussed its possible application to the solution to be developed. Although there were some interesting mechanisms that could be partially applied to this project, it was concluded that, to the best of our knowledge, there isn't at the moment a solution that completely meets the requirements for this project. Therefore, a new solution needs to be developed in order to provide intelligent load-balancing and multipath capabilities to the SITMe communication system.

Some of the existing tools to estimate the available bandwidth of a network path were studied as well. It was concluded that, in comparison to other tools, the best option to be used in the solution to be developed would be Yaz, which is an open source available

bandwidth estimation tool written in C/C++. Being the same programming languages as the ones used in the development of the prototype for this dissertation, it makes it easier to integrate in the developed solution.

Chapter 3

Multipath Transfer Solution

This chapter presents the proposed solution for the problem that motivated this dissertation.

The main goal of this project is to develop a multipath transfer solution that can be integrated in the SITMe project, making use of the available resources to provide intelligent traffic distribution and load-balancing, higher aggregated bandwidth, fault-tolerance, better QoE[5] and minimize financial costs of operation.

As explained in Section 2.1.1, the RBridges of the SITMe system are able to connect to public and private networks, being the public links mainly used to access the Internet through the Core RBridge (Internet gateway server), and the private links for peer-to-peer communications within the SITMe network.

From the pilot of the SITMe project running in a real environment it was observed that almost all the traffic originated from the bus passengers was for the Internet, which means that the majority of the traffic in the SITMe network was transferred over public links from RBridges to Core RBridge, and vice-versa. Therefore, the solution developed in this dissertation will be focused in the multipath exchange of data over public links between the Client Module and Server Module (described in Section 3.2).

3.1 - Requirements

It was established some requirements for the project to be developed, taking into consideration some of the requirements already established for the SITMe project. These requirements are the following:

- The design and implementation of the solution developed in this dissertation should respect the SITMe system architecture, by not making any significant changes in its operating system and network architecture;

[5] Quality of Experience

- The solution should have minimum cost, which means it should be as simple as possible, with the least amount of hardware and software required in addition to what is already used in the SITMe project;
- If possible, it should be used hardware and software already available in INESC Porto;
- The multipath transfer solution should:
    - Support multiple wireless network technologies. It should be supported at least one Wi-Fi interface and one UMTS interface in the bus module;
    - Provide load-balancing over multiple paths, by spreading the traffic over the available network interfaces in an intelligent way, based on metrics: available bandwidth, Round-Trip Time, jitter, packet loss, financial cost and type of traffic;
    - Support transmission of data over public links, by encapsulating the MPLS header with a standard transport layer protocol to be compliant with ISP's restrictions;
    - Cause minimum delay and packet loss in data flows;
    - Be generic enough so that it can also be easily adapted to other heterogeneous network systems.

3.2 - System Architecture

The system consists of two main software modules: the Client Module (integrated in the RBridge) and the Server Module (integrated in the Core RBrige). Since the traffic generated by the bus users is mostly bidirectional, both modules will act both as a sender and as a receiver. In this document, these two modules will, sometimes, be simply referred to as Client and Server.

Client Module: All the traffic generated by the bus passengers destined to the Internet is forwarded to the Client Module which, in turn, encapsulates it and sends it, over its wireless interfaces, to the Server Module located in the Core RBridge.

For the sake of simplicity, the Client Module will only support, at this stage, two wireless interfaces: A UMTS interface and a Wi-Fi interface.

It is assumed that, since these interfaces use two different technologies and connect to networks with very different characteristics, it should be enough to validate the good operation of the proposed solution, which provides data transferring over multiple paths with different properties.

Server Module: The Server Module will only have one physical Ethernet interface that will be associated with two virtual network interfaces, each with its own static IP address, that will be used to connect to each interface of the Client Module. The reason for this approach is to guarantee that all the traffic scheduled by the Client Module to be sent by each network interface, will effectively be sent by the intended interface. This is achieved by configuring, at the Client Module, the network gateways for each IP address of the Server Module.

The main role of this module is to forward the traffic that comes from the Client Module to the Core RBridge (and vice-versa), which in turn acts as an Internet gateway for the system.

Both modules will support simultaneous usage of its network interfaces, provided by an intelligent algorithm that spreads the data over the two interfaces to provide intelligent load-balancing, higher aggregated bandwidth and fault-tolerance to the network.

This intelligent traffic distribution and load-balancing over multiple paths is based on defined metrics: available bandwidth, RTT, jitter, packet loss, financial cost and type of traffic. These metrics, as well as the mechanisms used to estimate them will be further explained in Section 3.3 and Chapter 4.

A high level illustration of the architecture of the system is presented in Figure 3.1.

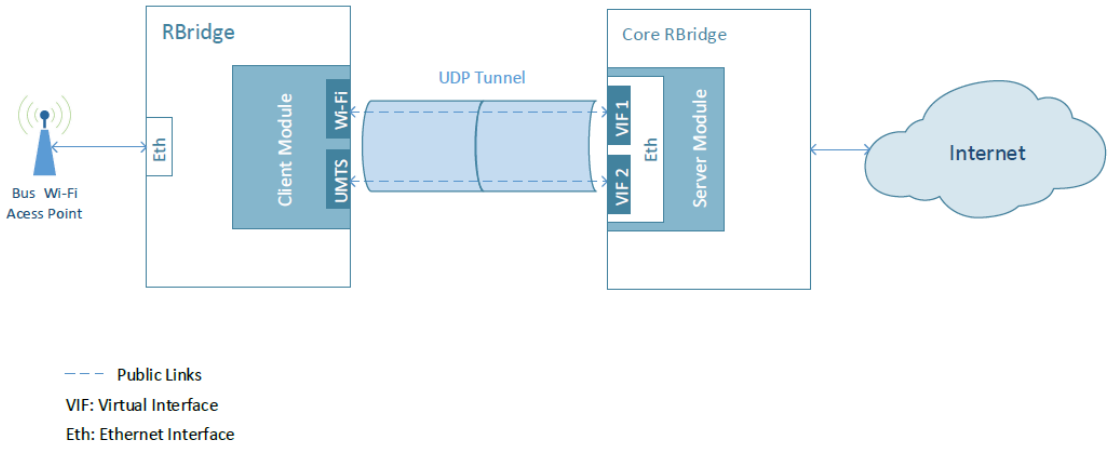

Figure 3.1 - System architecture of the Multipath Transfer Solution

The bus passengers connect to the system network through the Wi-Fi Access Point present on the bus, which is connected to the RBridge. The MPLS frames are then forwarded to a TAP interface, which is a virtual-network kernel device that simulates a link layer device, that receives the data from kernel-space and delivers it to a user-space program (the Client Module, in this case) which is attached to the device. This makes it possible to create a tunnel between the Client and Server Modules, by getting the original MPLS frames from the TAP interface and send them to the other end, encapsulated with a standard transport layer protocol.

As mentioned above, the Client and Server Modules act both as a sender and as a receiver. The data packets passed to the sender application by the TAP interface are sent to the receiver by distributing them in an intelligent way over the two network links and then aggregated at the receiver, to be forwarded to the its TAP interface. The packets are then forwarded to the Internet gateway, if the Server Module is the receiver, or to the bus passenger terminal, if the receiver is the Client Module.

Since there could be multiple Client Modules connected to the Server Module, there has to be some way for the Server to distinguish the incoming traffic from the Internet, so that it knows to which Client Module it belongs to, in order to forward it to the corresponding RBridge.

One way to do this would be to read the MPLS header of the incoming data packet to get the RBridge ID, but that would make this solution only suitable for MPLS networks. Since one of the requirements for this project is that the solution is generic enough so that it can also be easily adapted to other network systems, a different approach must be used. The basic idea of the solution for this problem is having the Server Module configure a TAP interface for each Client Module that connects to the Server. This way the Server knows that all the traffic coming from a certain Client Module must be forwarded to the corresponding TAP interface, and all the traffic coming from the Internet, captured from each TAP interface, must be forwarded to the corresponding Client Module. This is illustrated in Figure 3.2.

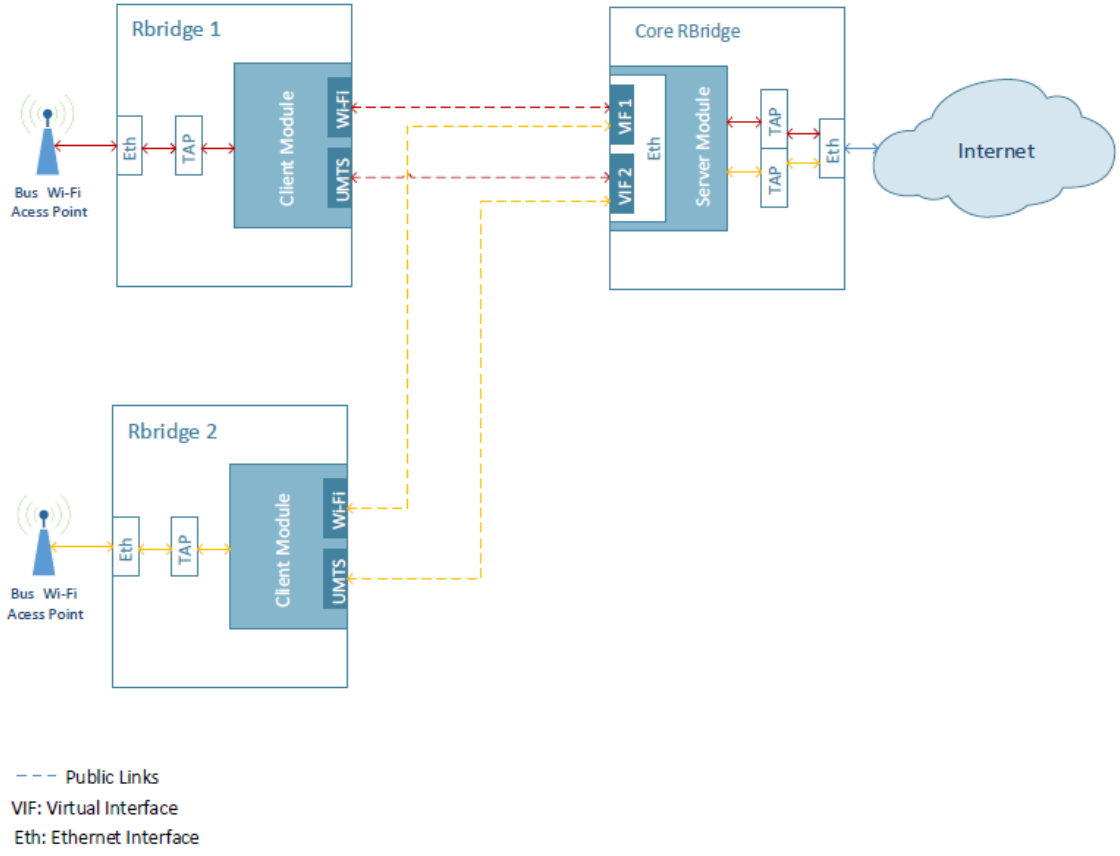

Figure 3.2 - System Architecture of the Multipath Transfer Solution for more than one Client Modules

3.3 - Metrics

In order to maximize the performance of the system and financial cost of operation, it has to be taken into consideration some properties of the network links that affect the overall performance, such as the available bandwidth and packet loss, as well as the financial costs of usage of the network links. Hence, it was defined several metrics that will be used by the multipath packet scheduling algorithm to better distribute the data over the available network links.

3.3.1 - Available Bandwidth

The bandwidth of a network link is the amount of data that it can transfer, from source to destination, per unit of time. In a multipath transfer solution it is important to have an estimate in real-time of the available bandwidth of each path in order to better balance the traffic load over the available paths. To estimate this metric it was used an open source available bandwidth estimation tool called Yaz, presented in Section 2.3, by adapting its source code, written in C/C++, to be integrated in the developed prototype.

3.3.2 - Round-Trip Time (RTT)

The Round-Trip Time is the time it takes for a packet to travel from source to destination plus the time it takes for an acknowledgement to that packet to be received at the source. This is a useful metric to be used by the packet scheduling mechanism to decide which links should be used to transfer delay sensitive traffic.

3.3.3 - Jitter

Jitter refers to the variation in packet delay, which can be a serious issue especially to real-time applications such as VoIP, that requires a steady stream of data to work properly. This metric is particularly useful for distributing the traffic produced by such applications in a way that optimizes their performance.

3.3.4 - Packet Loss Ratio

Packet loss occurs when one or more packets sent over a network link fails to reach its destination. Since UDP doesn't have a mechanism to keep track of the lost packets, it will be used a sequence number for the packets sent by each network interface that will be carried in the Control header (explained in Section 3.5), which will make it possible to keep track of the packets that fail to reach its destination or arrive too late, and calculate the packet loss ratio of the link.

3.3.5 - Economical cost

The economical cost is the monetary cost of usage of a particular network link. UMTS links have a higher cost than public Wi-Fi links, which usually are from Wi-Fi networks ceded to the SITMe project, free of charge, by Porto Digital.

3.3.6 - Type of Traffic

There are certain properties of the different technologies available to the system that should be considered in order to maximize its performance. For instance, while UMTS links have higher geographical coverage, providing higher availability and stability as the bus moves through the city, public Wi-Fi networks usually have short range, which means the Wi-Fi interface will be connecting to different networks, hence changing its IP address frequently. There will also be some sections of the bus course that will have no Wi-Fi coverage at all. In the other hand, Wi-Fi links usually have higher bandwidth and lower financial costs than UMTS links.

These properties should be taken into consideration in order to provide a better experience to the bus passengers, depending on the type of application they are using. For instance, a bus passenger making a VoIP call expects the call to have minimum delay and interruptions. Hence, the ideal technology for this type of traffic should be UMTS.

Having this in mind, it was defined four types of traffic generated by the users:

REAL-TIME: Traffic generated by real-time applications, such as VoIP and media live streaming. These applications are delay-sensitive and expect a steady stream of data in order to avoid unwanted interruptions.

INTERACTIVE: Traffic generated by interactive applications like web browsing. This type of traffic is also delay-sensitive, since it is generated by interactive applications, and therefore the user expects a response from the application in the least time possible.

DOWNLOAD/UPLOAD: Traffic generated by bus users downloading/uploading files. This type of traffic is more tolerant to delayed packets than the other defined types of traffic, since there is no real-time interaction required.

OTHER_TRAFFIC: Traffic that doesn't fit the other established types of traffic.

3.3.7 - REAL-TIME Cost

This metric is used to establish the network interface that should be preferred for transmitting REAL-TIME traffic. The lower the cost, the higher the preference to use the corresponding network interface. An equal value for both interfaces means that there is no preference.

3.4 - Data transportation over public links

SITMe system operates on a layer 2 overlay network implemented with MPLS protocol, as shown in Section 2.1.1, making it possible to have all the RBridges and the Core RBridge in the same virtual private network. Since it is not possible to transmit data encapsulated with MPLS header over public links due to restrictions imposed by ISPs, all the traffic exchanged between the RBridges and the Core RBridge over public links (e.g., UMTS) has to be encapsulated with a standard transport layer protocol.

The study of the available transport protocols, in Section 2.2, led to the conclusion that, to the best of our knowledge at the time, there is not a solution that completely meets the requirements for this project, therefore it was decided to implement a solution that uses a standard UDP header to encapsulate the original MPLS frames to transport them over public links, and to develop a Control header, which will be further explained in Section 3.5, that carries information such as the sequence number of the packet. The communication stack for this solution is presented in Figure 3.3.

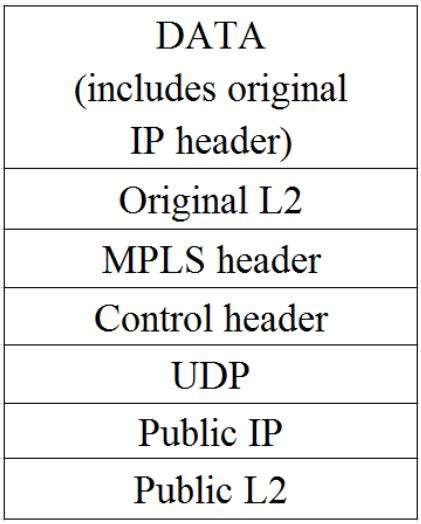

Figure 3.3 - Communication stack of the Multipath Transfer Solution

3.4.1 - UDP encapsulation

We decided to use the UDP protocol as the transport protocol used to encapsulate the MPLS frames, which was required in order to exchange data between the Client Module and the Server Module over public links, for reasons already explained in Section 3.4.

The reasons which led to the decision to use this protocol for this purpose are:

- UDP adds little extra overhead (8 byte header) [6] when compared to other transport protocols such as TCP (20 byte header) [5];
- UDP is more suitable to encapsulate TCP and UDP traffic than TCP, because, as explained in Section 2.2.1, tunneling UDP traffic over TCP would defeat the purpose of the UDP protocol, and tunneling TCP over TCP usually degrades the goodput of end-to-end TCP flows;
- UDP is simpler and easier to implement, since it is a connectionless protocol, which makes it possible, for instance, to use the same socket to receive data from multiple sources;
- The SITMe project already uses, with favorable results, UDP encapsulation to transport data over public links.

3.5 - Control header

The Control header carries information that will be used by the system to provide a better overall performance. This header consists of 4 fields (Sequence Number, Interface Sequence Number, Type of Traffic and Transport Protocol) which introduce an extra overhead of 72 bits (32bits for the Sequence Number; 32 for the Interface Sequence Number; 4 bits for the Type of Traffic field; 4 bits for the Transport Protocol field).

Sequence Number: Since UDP doesn't guarantee ordered delivery of transmitted packets, it is used a sequence number from a sequence shared by all the packets that are sent from or to a Client Module, which is different for each module in the Server to Client direction. This provides a way to keep track of the order of the packets of a given data flow and use it for reordering those packets at the receiver module (client or server). This is important because, as explained in Section 2.4.2, transmitting data packets over multiple paths with very different properties causes the transmitted packets to experience different delays, and, as a result, to be delivered with a different order with respect to what the source has sent. This sequence number is carried in a 32 bits field, which provides a sequence from 0 to 4294967295 (2^{32} -1).

Interface Sequence Number: UDP doesn't have a mechanism to keep track of the lost packets, therefore each data packet carries in its Control Header a sequence number that is different for each network interface, that allows the system to keep track of the packets who fail to arrive at the destination and to calculate the packet loss of the packets sent by each network interface. This field is also 32 bits, hence the sequence goes from 0 to 4294967295.

Type of Traffic: As explained in Section 3.3.6, there are different types of traffic that can be generated by the bus users. It is useful for the receiver to know which type of traffic the packet is transporting, so it can process it differently according to its properties. This means, for example, having different buffers with different holding times (the time the packets are held in the buffers to be reordered) to better process each type of traffic.

To introduce little overhead, it was used only a 4 bit field, which provides 16 (2^4) different values of information. The first 4 values will be used to identify the 4 types of traffic already established in this solution (0x00: OTHER TYPE; 0x01: REAL-TIME; 0x02: INTERACTIVE; 0x03: DOWNLOAD/UPLAOD). The remaining 12 values can be used in the future for other specified types of traffic.

Transport Protocol: This parameter refers to the transport protocol used in the original packet for the end-to-end communication. The motivation for including this parameter in the Control Header is similar to the Type of Traffic parameter. Although this information could be obtained by the receiver by removing the outer headers to inspect the original transport protocol header, the inclusion of this parameter in the Control Header enables the receiver to obtain that information with less processing, hence less delay.

As in the Type of Traffic parameter, the transport field has 4 bits. In this solution it was only considered two transport layer protocols: TCP and UDP. Other protocols are designated as OTHER PROTOCOL. The first 3 values of the field are used to identify these protocols (0x00: OTHER PROTOCOL; 0x01: TCP; 0x02: UDP). The remaining 13 values can be used in the future for other protocols.

Figure 3.4 presents a better view of the Control Header structure.

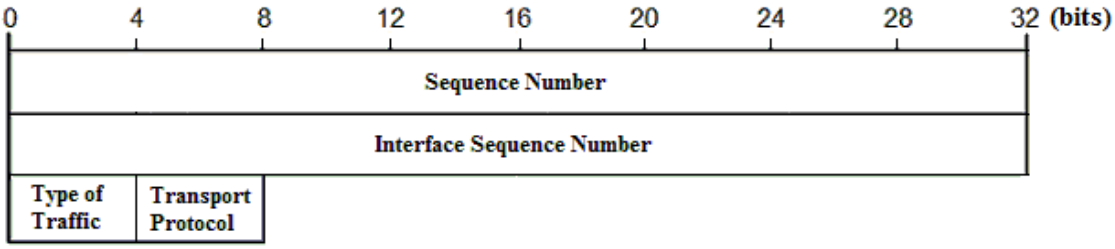

Figure 3.4 - Control Header structure

3.6 - HELLO System

Due to bus mobility, the wireless network interfaces (specially the Wi-Fi interface) of the Client Module will be frequently connecting to different networks, hence changing its IP address. Therefore, the Server Module needs to have updated information about the available network interfaces and their public IP addresses, in each bus, in order to connect to them.

To provide this information to the Server Module, it was implemented a mechanism (the HELLO System), that is responsible for registering and updating information related to the network interfaces that are used by each Client Module to transfer user generated network traffic from Client to Server Module and vice versa, such as its IP addresses and UDP ports to be reached, as well as other information like the RBridge ID the network interface belongs to, or costs assigned to each interface.

This is achieved by exchanging a series of messages that are responsible for carrying that information, as well as keeping the system updated about the availability of the network links between the two modules, i.e., if there is successful bidirectional communication between the corresponding network interfaces of the Client and Server. These messages and the overall operation of the HELLO System will be further explained in Section 4.4.

Chapter 4

Implementation

Chapter 3 presented an overview of the Multipath Transfer Solution proposed to provide multipath capabilities to the SITMe project. This chapter presents a wider view of how the solution was implemented, by explaining the design decisions made and the approach used in the development of each part of the developed prototype.

4.1 - Design Decisions

The main goal of this project is to develop a multipath transfer solution that is capable of using multiple network paths with different properties to transfer the traffic generated by bus user from a Client Module to a Server Module and vice versa. For the sake of simplicity, it was decided that it would be considered only two network interfaces of different technologies (UMTS and Wi-Fi) for each Client Module, which we consider enough to test and validate the concept of load-balancing network traffic over network paths with different characteristics.

It is assumed that all the resources (network links) are available to the system and will be used simultaneously, even if one link has enough available bandwidth to satisfy the demand. The load balancing of a DOWNLOAD/UPLOAD flow is performed based on link congestion, which means that if there are no signs of congestion in none of the two links, the flow will be split in a 50/50 way over the two network interfaces.

However, the ideal approach would be to only use a single link (preferably the one with the least financial cost) until it is congested, and then start using both if the demand is too high for that link. This approach was not implemented since it was not a priority for this solution, but should be looked at in future work.

The traffic distribution is based on a per-packet approach. This means that for each packet that arrives at the sender, it is decided by which network interface it should be sent, based on the defined metrics, unless the packet belongs to the DOWNLOAD/UPLOAD type of traffic. In that case, rounds of a defined number of packets are pre-scheduled to be distributed over the two interfaces, based on the available bandwidth of each network link.

The developed prototype was implemented in C/C++ programming languages, using the Integrated Development Environment (IDE) Eclipse CDT, in a Linux Operating System (OS) Ubuntu 14.04 LTS. C/C++ was chosen mainly because it provides better performance and lower level network programming and data manipulation, when compared to other programming languages. As for the used OS, it was chosen because Unix-like operating systems provide low level network traffic manipulation capabilities when compared to other operating systems such as Windows. In addition to that, it was also the Linux distribution already used in the SITMe project.

4.2 - UDP Tunneling

As explained in Section 3.4, an UDP tunnel needs to be implemented to be able to transport the original Layer 2 frames. This was achieved by using TAP interfaces, which are virtual-network kernel devices, that receive the data from kernel-space and delivers it to a user-space program (the Client Module, in this case) which is attached to the device. By capturing the original L2 frames from the TAP interface, at the sender, encapsulating them with the UDP protocol and sending them to the receiver, which, in turn, desencapsulates the received packets and forwards them to the its TAP interface, it makes it possible to have two TAP interfaces connected over the Internet as if they were on the same LAN.

As mentioned in Section 3.2, in order to separate the network traffic belonging to each Client Module, a tunnel is created for each client-server association. This means that each time a new Client Module registers itself on the Server, it is configured a new TAP interface on the Server associated with that Client. This way the Server knows that all the traffic coming from a certain Client Module must be forwarded to the corresponding TAP interface, and all the traffic coming from the Internet, captured from each TAP interface, must be forwarded to the corresponding Client Module. This feature is not completely implemented in the current version of the prototype, since it wasn't a priority for this dissertation, but should be implemented in future work.

In the current version of the developed prototype, each pair of TAP interfaces (one at the Client and the other at the Server) are preconfigured with IP addresses belonging to the same network, e.g., 10.0.0.1/24 and 10.0.0.2/24. Each module then attaches to the corresponding device by first opening, for read and write, the */dev/net/tun* device, which is the starting point for the creation of any TAP or TUN interfaces, and then uses its file descriptor and other parameters, such as the device name and type (TUN or TAP), as arguments of the *ioctl()* system call, which concludes the configuration and attachment to the TAP interface. The software modules are then able to read and write to the TAP interface using its file descriptor as if it was a normal Unix file.

4.3 - Control Messages

The developed system consists of series of independent modules that need to communicate with each other over a network to perform tasks such as registering the Client Modules on the Server and estimating the link metrics. In order to provide this kind of communication a series of control messages were designed to be used by the system's

mechanisms, such as the RTT or packet loss estimation mechanisms. All the control messages have the same basic structure:

MPTS|Message Number|Message Information|

MPTS: Stands for Multipath Transfer Solution, and is used to identify a message belonging to the control plane of this system. It is 4 characters long, followed by the "|" character (5 bytes in total), which should be enough to distinguish these messages from other random data.

Message Number: Is the number of the message used to identify the message type.

Message Information: Is the information carried by the message. This field can be different for each message type and will be further explained in the corresponding section of each message type.

These messages are exchanged over UDP sockets, using different IP addresses and UDP ports for each network interface. These parameters are registered by each Client in the Server, using the messages of the HELLO System, that is further explained in Section 4.4.

The complete list of the control messages of the Multipath Transfer Solution can be found in Appendix A.

4.4 - HELLO System

Since this system is implemented in a mobility scenario where the Client Module network interfaces may frequently connect to different networks, and, as a consequence, changing their IP addresses, it is important to keep the Server updated about the available network interfaces and their public IP addresses, as well as other important information related to them, in each RBridge. This section presents the design and implementation of the mechanism called HELLO System, which provides the above mentioned features to the Multipath Transfer Solution.

4.4.1 - Control Messages

A series of control messages were designed to be sent by each network interface that are responsible for carrying the relevant information about them, such as the UDP ports to be reached for data transferring, as well as keeping the system updated about the availability of the network links between the two modules, i.e., if there is successful bidirectional communication between the corresponding network interfaces of the Client and Server. In order to accomplish that, it was defined three different control messages: INTERFACE_REGISTRATION, REGISTRATION_COMPLETED and RC_ACK.

4.4.1.1 - INTERFACE_REGISTRATION

This message is sent periodically by each network interface of the Client Module in order to register itself on the Server as well as keeping the Server updated about its current state. It provides information such as the RBridge ID, interface ID, and UDP port to be used for data transferring. The IP address and UDP port used for the control messages of the system is not included in the message because it is obtained by the Server by reading the IP and UDP headers of the received packet. This message has the following structure:

MPTS|1|RBridgeID|InterfaceID|DataPort|ABEcontrolPort|ABEprobingPort|FinCost|RTcost|

- **RBridgeID:** Is the ID of the RBridge whom the Client Module belongs to.
- **InterfaceID:** Is the ID of the network interface that sent the message.
- **Data Port:** Is the UDP port to be used to transfer data from the Server Module to the Client.
- **ABEcontrolPort:** Is the TCP port to be used by the available bandwidth estimation mechanism as the receiver port for exchanging control messages.
- **ABEprobingPort:** Is the UDP port to be used by the available bandwidth estimation mechanism as the receiver port for the probing streams.
- **FinCost:** Is the value for the Financial Cost metric.
- **RTCost:** Is the value for the REAL-TIME Cost metric.

4.4.1.2 - REGISTRATION_COMPLETED

This message is sent by the Server in response to the INTERFACE_REGISTRATION message to acknowledge that the Client Module's network interface has been successfully registered. The structure of this message is the following:

MPTS|2|

4.4.1.3 - RC_ACK

RC_ACK is an abbreviation of REGISTRATION_COMPLETED ACKNOWLEDGMENT, and is used by the Client Module to acknowledge the reception of the REGISTRATION_COMPLETED sent by the Server, which also confirms that there is communication in the Server to Client direction. This message has the following structure:

MPTS|3|

4.4.2 - Operation

The basic operation of the HELLO System is described in the following steps:

1. Each Client Module registers their network interfaces, by sending an INTERFACE_REGISTRATION message to the Server Module through each of its network interfaces.
2. The Server Module registers the new network interface or updates it, with the information obtained from the INTERFACE_REGISTRATION message, if that interface is already registered. It then responds with a REGISTRATION_COMPLETED message to the corresponding network interface of the Client Module;
3. After receiving the REGISTRATION_COMPLETED message, the Client changes the state of the corresponding interface to Online and then responds with an RC_ACK message to acknowledge the successful reception of the REGISTRATION_COMPLETED message;
4. The Server receives the RC_ACK message and changes the state of the corresponding interface to Online;
5. The Client Module refreshes this information by sending new INTERFACE_REGISTRATION messages periodically, in order to keep the Server Module updated about the available network interfaces and their corresponding parameters such as the public IP addresses and UDP ports. If a Client's interface doesn't receive a REGISTRATION_COMPLETED message after a defined time span, it assumes the connectivity of that link is lost and declares the interface Offline. The same goes for the Server, which changes the state of the corresponding network interface to Offline if it doesn't receive an RC_ACK message after the same time span.

Since UDP is an unreliable connectionless protocol, the Online and Offline states are important to prevent the system from sending data to an interface that is currently unreachable. Therefore, the defined time interval for each round of HELLO System's messages should be short enough to guarantee the least data loss possible, in case of an interface becomes unreachable, but big enough so that it adds the least load possible to the network. In the current solution we use time intervals of 1 second for each round, and 3 seconds as the timeout value used for declaring an interface offline, which are the values we feel are good enough to provide a good compromise between those two factors.

For a better understanding of the sequence of HELLO System's messages exchanged by the two different modules (Client and Server), Figure 4.1 presents a diagram that illustrates this sequence of messages.

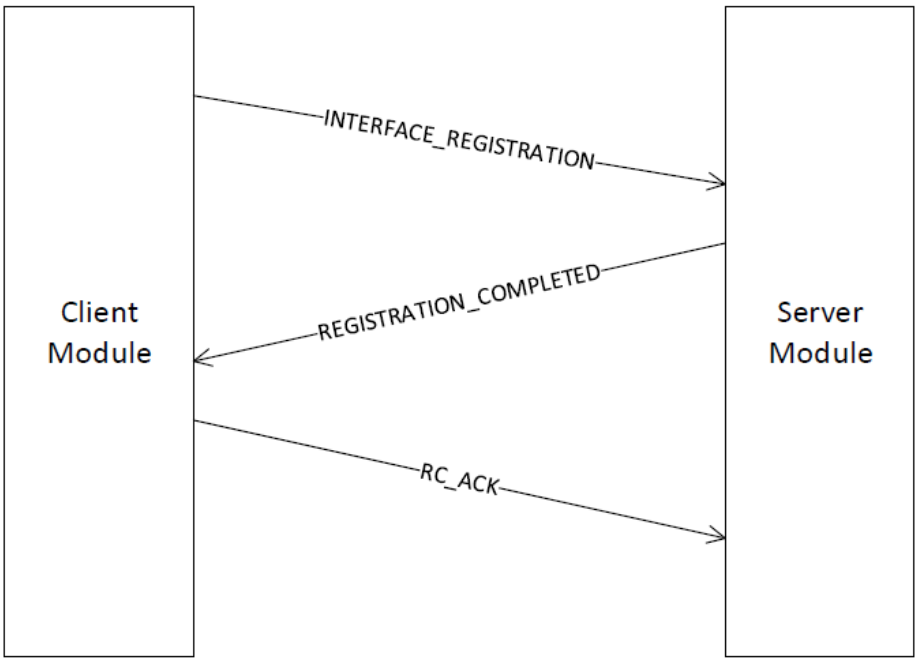

Figure 4.1 – Sequence diagram of the HELLO System's messages

4.5 - Link Metrics Estimation

As explained in Section 3.3, it was established a series of link metrics, such as the Available Bandwidth and Packet Loss Ratio, that are used by the Multipath Transfer Solution to better distribute the data over the available network links, as the link conditions change. Those conditions may change constantly, which requires some kind of mechanism that keeps estimating these metrics and providing feedback to the system about the link conditions. In order to achieve that, it was implemented three mechanisms for estimating the link metrics: One that is responsible for estimating the packet loss ratio, other for the available bandwidth and jitter, and another for the RTT.

Each module (Client or Server) only cares about the uplink conditions of their network links, hence these estimations are made from the sender to the receiver. The Client and Server can be both the sender and the receiver, depending on the direction of the network link being evaluated.

4.5.1 - Packet Loss Ratio

For estimating the packet loss ratio, it was implemented a mechanism that keeps track of the packets that are lost (or arrive too late) between each network interface of the sender and the receiver, using the Interface Sequence Number that is carried in the Control Header of each data packet.

4.5.1.1 - Operation

The basic operation of this mechanism for both the sender and the receiver is the following:

At the Receiver:

1. Every packet that arrives at the receiver has its Interface Sequence Number read and stored in the Received Packets List;
2. After a defined time, or if the list reaches a certain maximum number of elements, the system runs through this list to assess the numbers of the missing packets, and then clears the list;
3. The receiver sends a LOST_PACKETS message to the corresponding network interface of the sender, containing information such as the list of lost packets and the number of the first and last packets of the Received Packets List used to assess the missing packets;
4. If the system doesn't detect any missing packets on that list, it sends a NO_LOST_PACKETS message to the sender containing the number of the last packet of the Received Packets List.

At the Sender:

1. For each packet sent through each network interface it is added to its Control Header the corresponding Interface Sequence Number. The same number is stored in the Sent Packets List at the sender, so that it can be later used to estimate the packet loss;
2. When a LOST_PACKETS message arrives at a certain network interface, the system uses the information contained on this message to estimate the packet loss. This is done by comparing the numbers of the lost packets with the ones from the Sent Packets List, in a range that goes from the First Packet Number to the Last Packet Number (from the Received Packets List at the receiver), to determine which of the lost packets were effectively sent by this network interface. It then uses this information and the First Packet Number and Last Packet Number to calculate the packet loss ratio, and stores its value in the Packet Loss History List. It then clears the Sent Packets List up to the Last Packet Number.

   Having the sender calculate the packet loss through the procedures described above, instead of the receiver just reporting back the packet loss value, prevents, to a certain extent, false packet loss values that could be wrongly reported, caused by some system bug or message repetition, or even intentionally caused by an attacker, who could use the LOST_PACKETS message to report false packet loss values, thereby seriously degrading the performance of the system. The packet loss ratio is calculated using the formula presented in Equation 4.1;

$$Packet\ Loss\ Ratio = \frac{Number\ of\ Lost\ Packets}{Last\ Packet\ Number - First\ Packet\ Number} \times 100 \quad (4.1)$$

3. When a NO_LOST_PACKETS message is received, the system inserts a packet loss value of 0% in the Packet Loss History List, and then clears the Sent Packets List up to the Last Packet Number value of the list. This prevents the list from growing too big if a certain link doesn't experience packet loss for a long time span;
4. At every second, the system runs through the Packet Loss History List to assess the highest packet loss value, and uses it to update the Packet Loss Ratio metric of the corresponding network interface. This provides the worst case scenario for the estimated conditions of the network link during a 1 second time interval.

4.5.1.2 - Control Messages

In order to provide feedback to the sender about the link conditions regarding the occurrence of packet loss, it was designed two messages for this mechanism: LOST_PACKETS and NO_LOST_PACKETS.

4.5.1.2.1 - LOST_PACKETS

This message is sent from the receiver to the sender containing information such as the list of the lost packets and the first and last number of the range of packets investigated to determine the number of those lost packets. The structure of this message is the following:

MPTS|8|number_of_lost_packets|first_packet_number|last_packet_number|list_of_lost_packets|

number_of_lost_packets: Is the number elements contained the list of lost packets.

first_packet_number and last_packet_number: Is the number of the first and the last packet from the range of packets investigated to assess the missing packets.

list_of_lost_packets: Is the list of the numbers of the packets identified as lost by the receiver.

4.5.1.2.2 - NO_LOST_PACKETS

This message is sent by the receiver when there are no packets identified as lost when the Received Packets List is investigated for missing packets. This message has the following structure:

MPTS|9|last_packet_number|

last_packet_number: Is the number of the last packet from the range of packets investigated to assess the missing packets.

4.5.2 - Available Bandwidth and Jitter

In a multipath transfer solution it is important to have an estimate in real-time of the available bandwidth of each path, therefore a mechanism that provides this kind of feedback to the system needs to be implemented. Since the focus of this dissertation is not to develop a new bandwidth estimation mechanism, after evaluating the existing Available Bandwidth Estimation tools, we decided to use an open source tool called Yaz, presented in Section 2.3, by adapting its source code, written in C/C++. This allowed for an easier integration in the Multipath Transfer Solution, since it is the same programming language used in the development of the proposed solution.

This tool is based on the Probe Rate Model, which is based in the concept of self-induced congestion, already explained in Section 2.3. In order to estimate the available bandwidth, this tool uses probing streams that consist of trains of packets sent with a known spacing of time between them and then reports back the variations in those spacings. The jitter of a network link is the variation in the delay experienced by two or more consecutive network packets, and can be estimated by measuring the difference between the spacing of the packets at the sender and the spacing of those packets when they arrive at the receiver. Having this in mind, some modifications were made on Yaz's source code in order to be able to use the values provided by the control messages of the Available Bandwidth Estimation tool to also estimate the Jitter metric.

The Yaz tool consists of two separate software modules: the Sender and the Receiver. The available bandwidth is estimated in the direction from the Sender to the Receiver, i.e., the uplink available bandwidth from the Sender's perspective. This means that, to estimate the available bandwidth of each link in both directions, it needs to be running a Sender and a Receiver for each network interface of each Client Module, and another Sender and Receiver at the Server Module, for each corresponding Client interfaces registered in the Server. In order to achieve that, it is used 2 threads for each network interface of each module, one running the Sender and the other running the Receiver, which makes a total of 4 threads for each Client (2 threads per interface times 2 interfaces). The Server Module, in turn, has to have a total number of threads equal to 4 times the number of Clients connected to it (2 network interfaces per Client times 2 threads per interface).

The probing streams sent by the Sender introduce some load in the network link, which influences the estimation of the available bandwidth being made in the opposite direction. In order to avoid this issue, it was implemented a mechanism that makes the Client and Server take turns for each measurement round. This is achieved by having each pair of network interfaces communicate the beginning and end of each turn, in order to avoid unwanted traffic in the opposite direction. This communication is provided by a series of control messages that were designed for this purpose.

4.5.2.1 - Control Messages

In order to implement the turn-based mechanism mentioned above, it was designed 4 control messages: STARTING_PROBING, STARTING_PROBING_ACK, PROBING_DONE and PROBING_DONE_ACK. The purpose and structure of these messages is presented in the following subsections.

4.5.2.1.1 - STARTING_PROBING

This message is sent by the sender when it starts a new round of available bandwidth estimation. This message has the following structure:

MPTS|4|

4.5.2.1.2 - STARTING_PROBING_ACK

This message is sent by the receiver to acknowledge the reception of the STARTING_PROBING message. This message has the following structure:

MPTS|5|

4.5.2.1.3 - PROBING_DONE

When a measurement round is concluded the sender sends this message to inform the receiver that its turn has finished, which enables the other module to start a new measurement round in the opposite direction. This message has the following structure:

MPTS|6|

4.5.2.1.4 - PROBING_DONE_ACK

This message is sent by the receiver to acknowledge the reception of the PROBING_DONE message. This message has the following structure:

MPTS|7|

4.5.2.2 - Operation

The basic operation of the mechanism responsible for estimating the Available Bandwidth and the Jitter metrics is described in the following steps:

1. Each network interface has a Boolean variable, called Probing Turn, that indicates if it is that interface's turn to do the probing round. When it is the turn of a certain network interface to start a probing round (Probing Turn value is 1), it sends a PROBING_STARTING message to the corresponding receiver interface.

2. The receiver receives the PROBING_STARTING message and updates its Probing Turn variable to 0, if it is not already. It then sends a PROBING_STARTING_ACK to confirm that the PROBING_STARTING message was successfully received, and to assure that it will not proceed with another probing round in the opposite way until it receives a PROBING_DONE message.
3. The receiver estimates the available bandwidth and the jitter of the network link, based on the Yaz's control messages reported back from the sender, and updates its values in the system. When the probing round is concluded, the sender sends a PROBING_DONE message to the corresponding receiver interface.
4. After receiving the PROBING_DONE message, the receiver updates the Probing Turn variable of the corresponding interface to 1, if it is not already. It then sends a PROBING_DONE_ACK message to acknowledge the reception of the PROBING_DONE message.
5. When the sender interface receives the PROBING_DONE_ACK message it changes its Probing Turn variable to 0. If it doesn't receive PROBING_DONE_ACK, it keeps its Probing Turn variable at 1, and proceeds to start a new measurement round. This allows the sender to keep estimating the available bandwidth, even if there is some problem at the other end that is momentarily preventing the receiver from sending the PROBING_DONE_ACK message.

4.5.3 - Round-Trip Time (RTT)

The RTT, commonly known as ping time, is the time it takes for a packet, sent by the sender, to arrive at the receiver, plus the time it takes for the response to that packet, sent by the receiver, to arrive at the sender. This is a useful metric to be used by the Multipath Transfer Solution to decide which paths should be used to transfer delay sensitive traffic. In order to estimate this metric, it was implemented a simple mechanism that measures the time difference between the moment a message is sent by the sender and the moment it receives the response from the receiver. Although there is already an existing network utility called Ping that uses ICMP echo messages to provide this kind of estimation, we decided to implement our own mechanism, since it was easier to implement than to integrate the Ping utility in the developed prototype.

4.5.3.1 - Control Messages

To implement the RTT estimation mechanism it was designed two control messages: PING_REQUEST and PING_RESPONSE. The purpose and structure of these messages is presented in the following subsections.

4.5.3.1.1 - PING_REQUEST

This message is sent by the sender to request a response from the receiver, and has the following structure:

MPTS|10|SequenceNumber|

4.5.3.1.2 - PING_RESPONSE

This message is sent by the receiver as a response to the PING_REQUEST message received from the sender. It has the following structure:

MPTS|11|SequenceNumber|

4.5.3.2 - Operation

The operation the RTT estimation mechanism is described in the following steps:

1. At each second, the sender gets the local time through the system call *gettimeofday()*, which returns the time of the local machine with microsecond precision. It then sends a PING_REQUEST message to the receiver. The timestamp and the corresponding Sequence Number of the PING_REQUEST message are stored in memory to be used later to calculate the RTT;
2. When the PING_REQUEST message arrives at the receiver, it immediately replies with a PING_RESPONSE message with the same Sequence Number as the received PING_REQUEST;
3. When the sender receives the PING_RESPONSE message, if it matches the Sequence Number of the last PING_REQUEST message, it gets another timestamp and calculates the difference between this timestamp and the one obtained just before the PING_REQUEST message was sent. The calculated value is the RTT for the corresponding network link.

4.6 - Packet Reordering

Transmitting data packets over multiple paths with different properties may cause the transmitted packets to experience different delays, and, as a result, to be delivered with a different order with respect to what the source has sent. To address this issue, it was implemented a mechanism to reorder the data packets exchanged between the modules of the Multipath Transfer Solution.

Each data packet that arrives at a certain module, coming from any network interface of that module, is stored in a buffer that holds the packets for a defined time to be reordered later. After that time, or if the number of packets in the buffer reaches a maximum value, the packets are reordered and flushed to the corresponding TAP interface. Different buffers

are used for each one of the different types of traffic. This allows the system to have different holding times (the time span between each flushing to the TAP interface) that better suit the properties of each type of traffic. For example, the DOWNLOAD/UPLOAD type is much less sensible to delay than the INTERACTIVE traffic and therefore it should have a longer holding time, since this allows for a larger group of packets to be reordered.

During the development of this mechanism, it was noticed that, when the thread responsible for the reordering of the packets of a certain type of traffic is performing the reordering and flushing routine on the corresponding buffer, there was some packet loss, because the system was unable to store in that buffer the packets that arrived during that time. In order to solve this problem, it was added an extra buffer to each type of traffic so that one buffer can store the incoming packets while the other is being reordered and flushed to the TAP interface. This routine is performed alternately so that there is always a free buffer to store the incoming data packets. Therefore, in the current solution there are 2 buffers for each type of traffic, in each Client Module, which makes a total of 6 (2 buffers times 3 types of traffic). This means that the Server Module will have 6 buffers times the number of Clients connected to it.

It is used one thread that is responsible for sending each incoming packet to the corresponding buffer, and also one thread, for each type of traffic, responsible for reordering and flushing the packets to the TAP interface. A high level illustration of the operation of these two threads is presented in Figure 4.2 and Figure 4.3. Please notice that the buffer 1 and buffer 2 in those figures refer to the pair of buffers corresponding to each type of traffic.

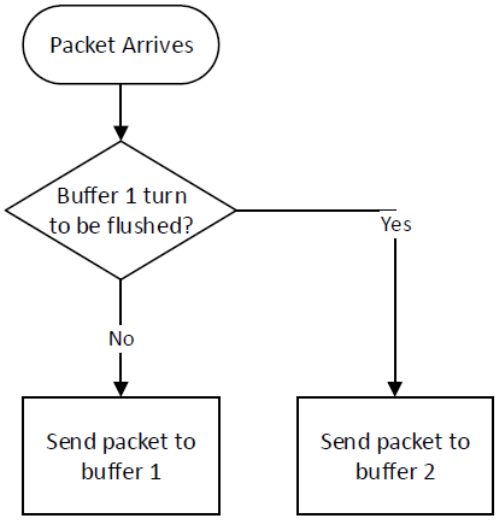

Figure 4.2 – Flowchart of the thread that handles received data packets

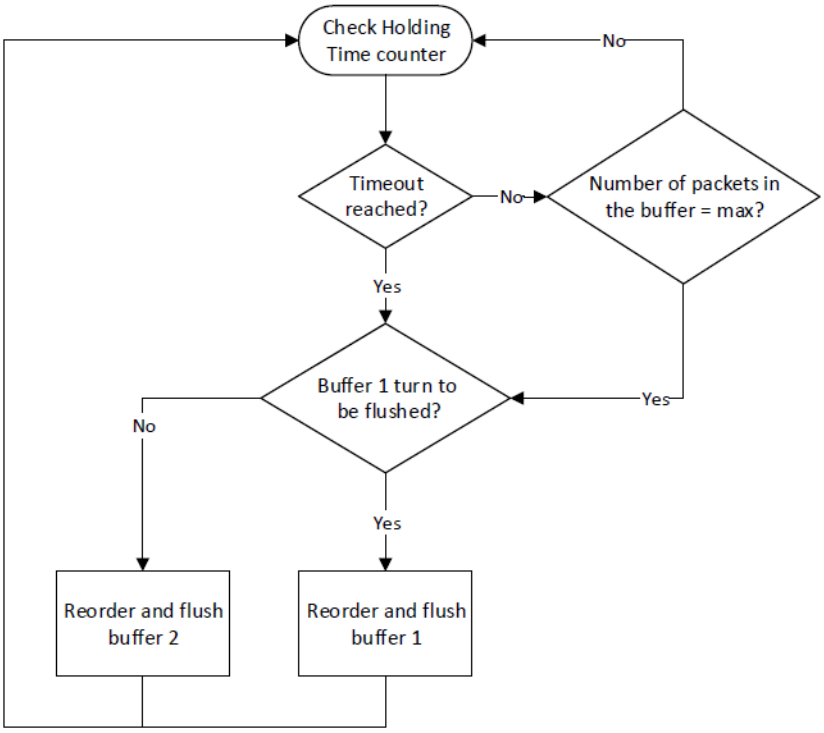

Figure 4.3 - Flowchart of the thread responsible for reordering and flushing packet buffers

4.7 - Type of Traffic Identification

The development of a mechanism that would be able to accurately identify each type of traffic for each packet to be sent would be too complex and time consuming for the scope of this dissertation. Therefore, we decided to use a simpler approach that, although would not be suitable for real life network environments, it allows us to distinguish the packets belonging to the 3 established types of traffic for testing purposes.

In order to distinguish the 3 types of traffic, it was defined that all UDP traffic is considered REAL-TIME traffic, and all TCP traffic is considered DOWNLOAD/UPLOAD traffic. The transport protocol of each packet is identified by reading the Protocol field of the IP header of each packet. If the protocol read is neither TCP nor UDP, the packet is designated as OTHER_TRAFFIC.

To identify the INTERACTIVE traffic, it was implemented a second TAP interface, in addition to the one used for the REAL-TIME and DOWNLOAD/UPLOAD traffic. All the traffic that comes from this TAP interface is identified as INTERACTIVE traffic.

In summary, all the packets that arrive to the sender module by TAP_0 are identified as REAL-TIME or DOWNLOAD/UPLOAD traffic, depending on its transport protocol (UDP and TCP, respectively). All the packets that arrive to the sender by TAP_1 are identified as INTERACTIVE traffic.

4.8 - Traffic Distribution and Load-Balancing

One of the main goals of this project is to provide intelligent traffic distribution and load-balancing over the available network links (based on defined link metrics and type of traffic), as well as higher aggregated bandwidth and fault-tolerance to the SITMe project's communication system. In order to achieve that goal, it was implemented a mechanism, based on a per-packet approach, that decides which network interface should be used to send each packet, based on its type of traffic and the conditions of the available network links.

As mentioned in Section 3.3.6, there are 3 defined types of traffic (REAL-TIME, INTERACTIVE and DOWNLOAD/UPLOAD). All the traffic that doesn't fit the properties of these 3 types is designated as OTHER_TRAFFIC. Since OTHER_TRAFFIC has unknown properties, in this solution it isn't currently considered for intelligent distribution and therefore it is just forwarded to one of the available network interfaces.

Packets identified as REAL-TIME and INTERACTIVE traffic are scheduled to be sent by the best available network interface for the corresponding type of traffic at the moment the packet is to be sent. Hence, it could be said that, although each type can be sent by different network interfaces at the same time, there is no real load-balancing for those two types of traffic. This is mainly due to the fact that REAL-TIME traffic is preferred to be sent by the network link with the least REAL-TIME cost, and the INTERACTIVE type, considering its delay sensitive properties, benefits more from using the link with the least delay possible.

The DOWNLOAD/UPLOAD type of traffic, in the other hand, has different properties, such as higher bandwidth demand and much lower delay and jitter sensitivity, and therefore is more suitable to be load-balanced over the available network links with different properties, in order to provide a higher aggregated bandwidth. This is achieved by pre-scheduling rounds of a defined number of packets to be distributed over the two interfaces, based on link congestion and other metrics.

Each DOWNLOAD/UPLOAD packet is sent to an outgoing buffer where the packets wait to be sent through the intended network interface, at a rate set by the Current Packet Spacing of the corresponding network interface. The Current Packet Spacing is the time value in microseconds that defines the time spacing between each sent packet, and is dynamically adjusted based on the corresponding link congestion.

4.8.1 - Estimating link congestion

In order to be able to implement a load-balancing mechanism that distributes the data packets over the available network links, it is important to have feedback in real-time of the current congestion of those links. Ideally, this feedback would be provided by the available bandwidth estimation mechanism, however, during the development of this solution, it was noticed that the used Available Bandwidth Estimation tool is not able to provide such feedback with the required frequency and accuracy, which made it unsuitable to be used as a metric for load-balancing. The frequency and accuracy of the estimations may be enhanced by tweaking the configuration parameters of the Yaz tool, but this results in a significant increase in the load added to the network links being probed, which may be an issue, especially in low bandwidth links. Therefore, we decided to use the available bandwidth

estimation mechanism to perform the probing rounds on a certain link, only when there is no significant traffic being transmitted over that link, and use the estimated bandwidth to calculate and adjust the Current Packet Spacing of each interface, so that when that interface starts to transmit data, the value of its Current Packet Spacing is already near optimal.

One indication of link congestion is the occurrence of packet loss. Hence, as an alternative to the Available Bandwidth metric, we thought of using the Packet Loss Ratio metric to provide feedback to the load-balancing mechanism. However, this alternative proved to be useless, since end-to-end traffic flows with congestion control mechanisms, such as TCP, are very quick to adjust the bitrate of their flows based on the network conditions, which leads to little to no packet loss observed on the network links being used that could indicate link congestion.

The third alternative, which is the one used in the current prototype, was to use the size of the UDP sockets send queues (i.e. the number of bytes in the queue waiting to be sent). After some investigation of the behavior of this queues, it was implemented a mechanism that uses this values (of the socket send queue corresponding to each network interface) to estimate the current congestion of each network link. The size of these queues is obtained using the *ioctl()* system call with the TIOCOUTQ argument.

However, this approach has some limitations. The UDP socket sends the packets at the maximum rate possible provided by the corresponding network interface available bandwith, and therefore the send queues will only grow, in a noticeable way, if the application (the Client or Server modules, in this case) is trying to send the packets at a rate superior than that maximum rate. Hence this method only works if the limitation on the network link bandwidth is caused by the sending speed of the network interface. If the bandwidth bottleneck is anywhere along the network path, this method becomes useless because the UDP socket at the sender will still be sending at the maximum rate possible, but, as mentioned above, the occupation of the send queues will not be as noticeable, and therefore this behavior cannot be used to assess the link congestion.

Nevertheless, since there is no better alternative at the moment, to the best of our knowledge, we decided to use this approach in the current version of the prototype. Although this method might not be suitable for real life network conditions, we feel it is good enough to be used to evaluate the good operation of the Load-Balancing algorithm. Nonetheless, in future work it should be implemented a new way to provide feedback to the Load-Balancing mechanism. The ideal solution would be to develop a new available bandwidth estimation mechanism that is able to provide estimations with the required accuracy and frequency without adding too much load to the network link.

4.8.1.1 - Adjusting the Packet Spacing

As mentioned in Section 4.8, the Current Packet Spacing of each network interface is adjusted dynamically based on the conditions of the network link. This feedback is currently provided by the mechanism that estimates the link congestion based on the socket send queues described in the previous section. The operation of this mechanism is described in the following steps:

1. During a period of 100 milliseconds, the system collects 10 samples of the current size of the send queue. It then calculates the Send Queue Size Above Max Ratio (SQSAMR), that consists of the number of samples above a defined maximum value (NSAM) per number of samples collected. This formula is presented in Equation 4.2.
2. If the SQSAMR is equal or above a defined value (0.5 in the current prototype), the Current Packet Spacing of the corresponding network interface is incremented. If not, the Current Packet Spacing is decremented.

$$\text{SQSAMR} = \frac{\text{NSAM}}{\text{Total number of samples}} \tag{4.2}$$

4.8.2 - Link Quality

For all the three types of traffic, the Packet Loss metric is the ultimate decider about which network interface should be used to send the packet, because a significant packet loss in a network link can seriously degrade the throughput of the end-to-end flows. Therefore, it was implemented a mechanism that evaluates if a link is constantly suffering from packet loss during a period of time, and determines the quality of that link based in that evaluation. The overall operation of this mechanism is the following:

1. At each second, the system collects the value of the Packet Loss Ratio of each network link. After collecting 5 values (for 5 seconds), it calculates the Packet Loss Above Max Ratio (PLAMR), which consists of the ratio between the number of times the Packet Loss Ratio metric for that link was above a defined maximum value (1% in the current prototype), divided by the total number of values collected, which is 5. If the PLMAR is higher than a defined value (60% in the current prototype), the network link is declared a Low Quality Link.
2. When a link is declared a Low Quality Link, it is only checked again after a defined time (20 seconds in the current prototype), i.e., after a link is declared a Low Quality Link, the routine that collects Packet Loss Ratio values and evaluates the quality of the link based on those values is only performed again after the defined time span. This prevents the system from keeping transmitting traffic over a link which is constantly suffering from packet loss during a long period of time. Since the Packet Loss Ratio estimation is based on the data packets sent over each network interface, if the system kept evaluating the quality of such a link with a 5 second time span, it would be losing more packets unnecessarily. By waiting a longer period for a new evaluation, it gives more time for the conditions of that link to eventually improve, while the traffic is being sent by another link with better conditions.

4.8.3 - Packet scheduling for REAL-TIME traffic

The most relevant metric for the REAL-TIME traffic is the REAL-TIME cost. The reason for this is because it was established that traffic generated by real-time applications, such as VoIP, should be preferentially sent by a defined interface, such as UMTS. The REAL-TIME cost is the way to let the system know which one of the network interfaces is to be preferred to send the REAL-TIME traffic. If the value is the same for both interfaces, then there is no preference, and the RTT is used as the most relevant metric. If the difference between the RTT of the two network interfaces is equal or lower than a defined maximum value (5 milliseconds in the current solution), the deciding factor is the Financial Cost of each interface. This is due to the fact that a small difference between the RTT of each interface is not significant for the QoE of real-time applications, and therefore it is used the interface with the lowest Financial Cost.

As explained in Section 4.8.1, the Available Bandwidth estimation mechanism is only used when there are no significant data being transmitted, and since the Jitter metric is estimated using that mechanism it can't be used as a decision metric for packet scheduling. For that reason, and since the jitter experienced by the data packets is attenuated anyway by the buffers used for reordering the packets at the receiver, it is not considered in the REAL-TIME packet scheduling algorithm.

The last deciding factor is the quality of the link in terms of packet loss. Even if a packet is scheduled to be sent by a certain link, based on the other metrics, if the link is considered a Low Quality Link at the moment the packet is to be sent, the packet is sent by the other link, if the other link is not a Low Quality Link as well. If the links of both interfaces are considered Low Quality Links, then the packet is sent by the one with the lowest Packet Loss Ratio. This is valid for the 3 types of traffic.

The flowchart that illustrates the algorithm of the REAL-TIME packet scheduler is presented in Figure 4.4.

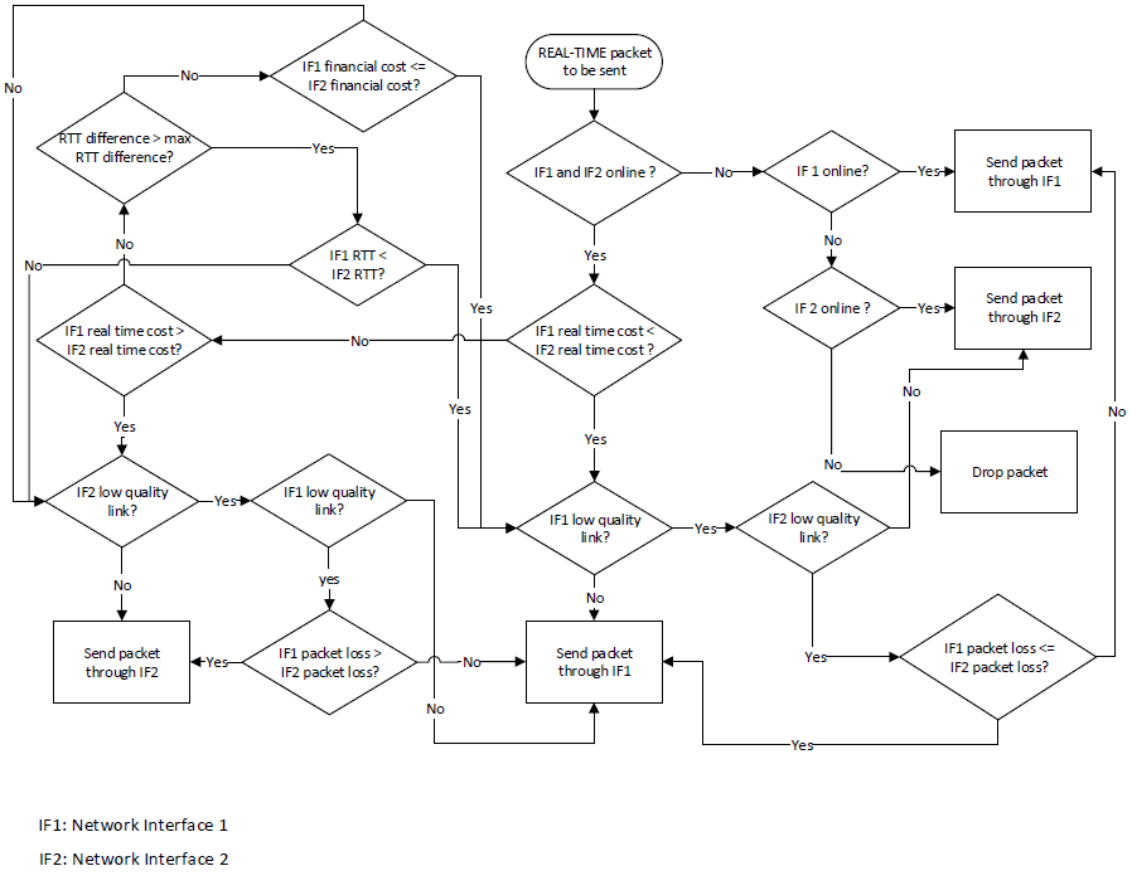

Figure 4.4 – Flowchart of the REAL-TIME packet scheduling algorithm

4.8.4 - Packet scheduling for INTERACTIVE traffic

The INTERACTIVE traffic is generated by delay-sensitive applications. Thus, the most relevant metric for this type is the RTT. As for the REAL-TIME traffic, if the difference between the RTT of the two network links is equal or lower than a defined maximum value (5 milliseconds in the current solution), the deciding factor is the Financial Cost of each interface. The rest of the algorithm is similar to the REAL-TIME packet scheduling algorithm, which has been already explained in the previous section. In Figure 4.5 it is presented the flowchart of the INTERACTIVE packet scheduling algorithm.

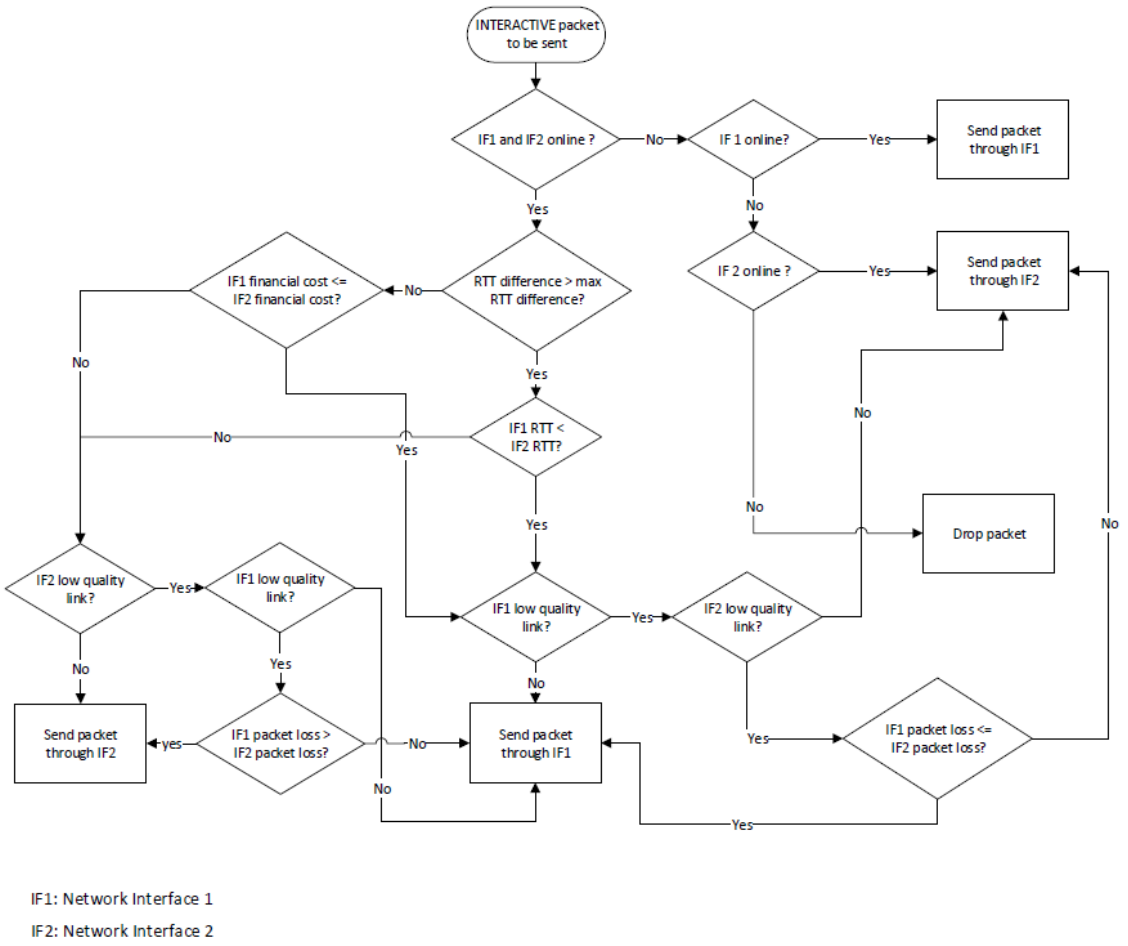

Figure 4.5 - Flowchart of the INTERACTIVE packet scheduling algorithm

4.8.5 - Packet scheduling for DOWNLOAD/UPLOAD traffic

As mentioned in Section 4.8, rounds of a defined number of packets are pre-scheduled to be distributed over the two interfaces, based on link congestion and other metrics. In the current solution these rounds consist of groups of 100 packets which are assigned to be sent by each interface based on the packet distribution, which is calculated based on the values of the Current Packet Spacing of each interface. The way this distribution is calculated will be further explained in Section 4.8.5.1.

In order to provide a better distribution of the packets, the scheduler divides the rounds of 100 packets in 10 or 11 sub rounds (depending on the packet distribution) and distributes the packets over those sub rounds according to the calculated packet distribution. This means that if, for instance, the calculated distribution is 75 packets for Network Interface 1 and 25 packets for Network Interface 2, the scheduler distributes the packets in 10 sub rounds of 9 packets (7 to be sent by Interface 1 and 2 packets to be sent by Interface 2) plus one sub round of 10 packets (5 to be sent by Interface 1 and 5 to be sent by Interface 2).

As mentioned in Section 4.8.3, the Packet Loss Ratio metric is the ultimate decider about which network interface should be used to send the packet. Hence, even if a packet is

scheduled to be sent by a certain interface, it will still be sent by the other interface if it offers better link conditions in terms of packet loss.

The overall algorithm for the DOWNLOAD/UPLOAD packet scheduler is illustrated in Figure 4.6.

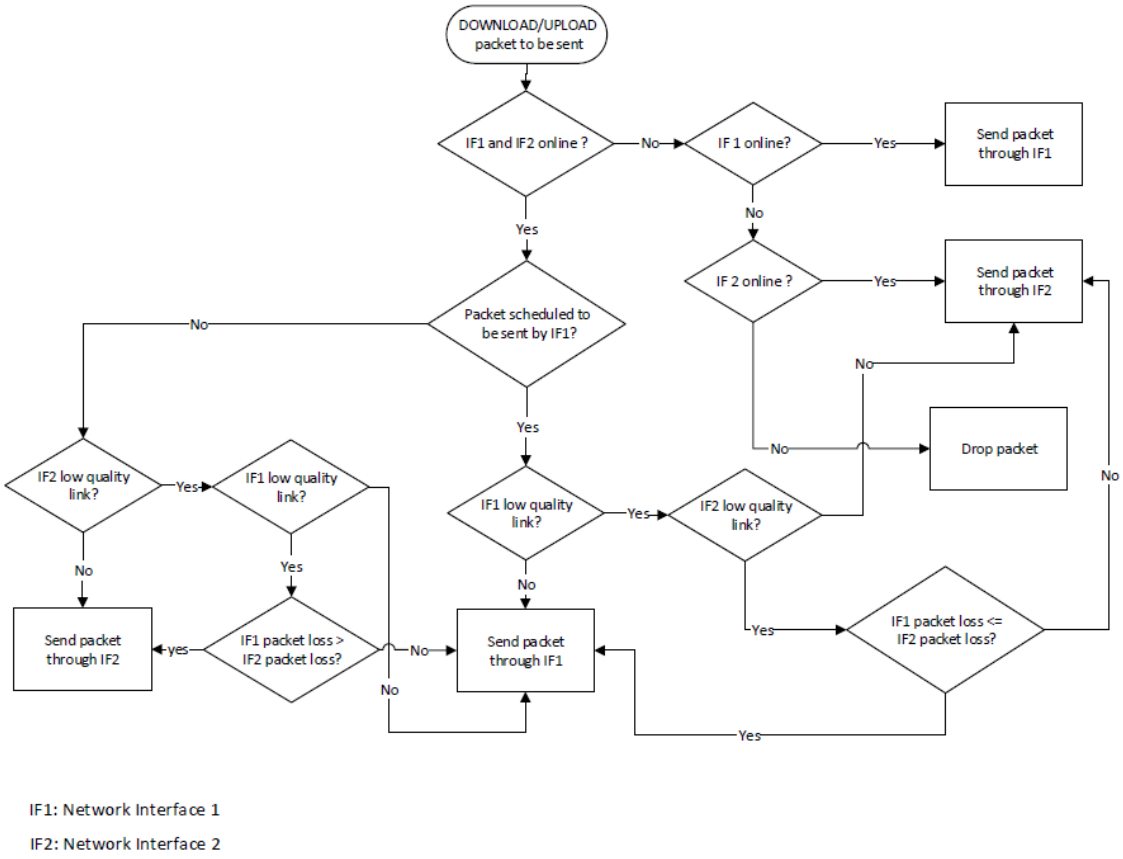

Figure 4.6 - Flowchart of the DOWNLOAD/UPLOAD packet scheduling algorithm

4.8.5.1 - Calculating Packet Distribution

The packet distribution is calculated based on the Current Packet Spacing of each network interface. As mentioned in Section 4.8, the Current Packet Spacing of each interface is dynamically adjusted based on the link congestion. Therefore, it provides a good indicator of the congestion of each link, which can be used to calculate the packet distribution that better suits the current conditions of the available links. This calculation is described in the following steps:

1. First it is calculated the ratio (α) between the Current Packet Spacing of Interface 1 (CPS1) and the Current Packet Spacing of Interface 2 (CPS2).
2. Then that ratio is used to calculate the number of packets, out of 100, that should be sent by Interface 1 (NP1) and by Interface 2 (NP2).

The formulas used for these calculations are presented in Equations 4.3, 4.4 and 4.5.

$$\alpha = \frac{CPS1}{CPS2} \tag{4.3}$$

$$NP1 = \frac{100}{\alpha+1} \quad (rounded\ to\ the\ nearest\ integer) \tag{4.4}$$

$$NP2 = 100 - NP1 \tag{4.5}$$

Chapter 5

Evaluation of the Solution

Chapter 3 and Chapter 4 presented the specification, design and implementation of the proposed Multipath Transfer Solution. In this chapter it is evaluated the performance of the implemented solution. This evaluation consists of a series of tests designed to individually evaluate the operation of the several mechanisms that constitute the Multipath Transfer Solution, as well as the overall performance of the system as a whole, by simulating the dynamic changes in the conditions of the networks the system uses to transfer the network traffic between the two modules (Client and Server).

5.1 - Testing Environment

Two different machines were used, running the Ubuntu 14.04 Operating System, each one with two Fast Ethernet network interfaces, which provide a nominal rate of 100 Mbit/s. One machine runs the Client Module and the other the Server Module.

The two corresponding network interfaces of each machine are directly connected by wire to each other, as shown in Figure 5.1. By directly connecting the interfaces, instead of using other networking devices, such as a router or a switch, it is guaranteed that there are no exterior factors influencing the link conditions during the tests.

In order to simulate the link conditions, NetEm [24] was used, which is a network simulation tool that uses the existing Quality of Service (QoS) and Differentiated Services facilities of the Linux Kernel to simulate network conditions, by adding delay, packet loss, and other characteristics, to the outgoing packets of a certain network interface. This tool, however, is not able to provide bandwidth simulation by itself, thus an additional tool called Token Bucket Filter [25] was also used, in order to provide bandwidth limitation to the outgoing traffic of a network interface.

In order to implement dynamic simulation scenarios, Bash (Unix shell) scripts were used, which were programmed to execute the commands of the above mentioned network simulation tools in order to dynamically change the conditions of the network links, as well as performing other operations in real-time, such as changing the IP addresses of the network interfaces or turning them on and off to simulate availability.

In order to generate traffic, it was used Iperf [26], which is a network testing tool that generates data streams (TCP and UDP), between a client and a server, and measures the throughput, as well as other parameters, such as the packet loss and jitter, of the network that is carrying them.

In this chapter the names of the 3 types of traffic may be abbreviated to RT, IA and DL/UL, for the REAL-TIME, INTERACTIVE and DOWNLOAD/UPLOAD types of traffic, respectively.

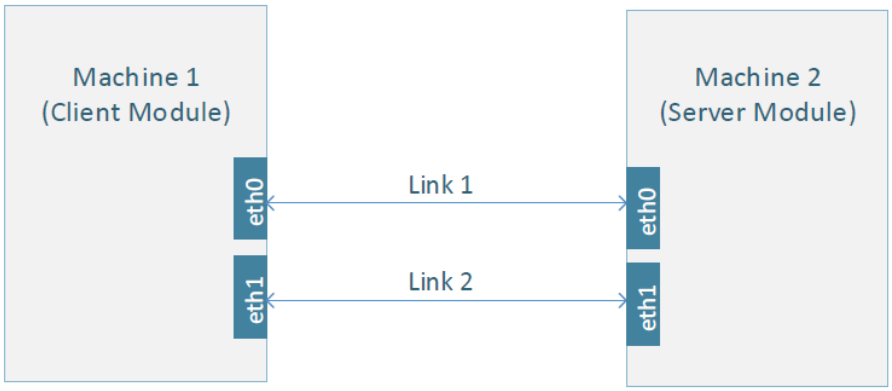

Figure 5.1 – Testing setup

5.2 - Isolated Tests

The Multipath Transfer Solution consists of a series of mechanism that depend on each other to provide intelligent traffic distribution and load-balancing to the system. For a better observation and evaluation of the behavior of the different mechanisms that compose the implemented system, as the network conditions change, a series of tests were designed to provide experimental data that can be used to validate the good operation of those mechanisms. Since the behavior of the Client and Server is similar in terms of data transferring and they can both act as a sender and as a receiver, in these tests, to avoid redundant testing, the system will be evaluated in only one direction, from Server to Client, i.e., the traffic flows will be generated using Iperf from Server Module to Client Module, where the Server is the sender and the Client is the receiver. The properties of each link are also in the Server's perspective, i.e., for example, unless stated otherwise, the bandwidth of link 1 is the uplink bandwidth of that link from the Server's perspective.

5.2.1 - Links with different available bandwidths

This test aims to evaluate the operation of the load-balancing algorithm as the available bandwidth of each network link changes. The simulation scenario for this test consists of 6 sections with different bandwidth conditions, separated by time intervals of 40 seconds.

5.2.1.1 - Test 1.1 - Performance with DOWNLOAD/UPLOAD traffic

In this test, we generated a DOWLOAD/UPLOAD traffic flow (TCP flow, using Iperf) from Server to Client and measured the outgoing bitrate of each network interface of the Server, and the TCP throughput measured by Iperf at the Client, as the available bandwidth of the links change during the test. These measurements can be seen in Figure 5.2. The simulation of the bandwidth changes in the network links are achieved by limiting the bandwidth of the network interfaces of the Server, using a Bash script that was designed for this purpose.

The characteristics of the 6 defined testing sections can be seen in Table 5.1.

		Bandwidth (Mbit/s)	RTT (ms)	Packet Loss (%)	Fin. Cost	RT Cost
Section 1 (0s to 39s)	**Link 1**	20	2	0	2	1
	Link 2	20	2	0	1	2
Section 2 (40s to 79s)	**Link 1**	20	2	0	2	1
	Link 2	10	2	0	1	2
Section 3 (80s to 119s)	**Link 1**	30	2	0	2	1
	Link 2	10	2	0	1	2
Section 4 (120s to 159s)	**Link 1**	15	2	0	2	1
	Link 2	35	2	0	1	2
Section 5 (160s to 199s)	**Link 1**	25	2	0	2	1
	Link 2	25	2	0	1	2
Section 6 (200s to 239s)	**Link 1**	20	2	0	2	1
	Link 2	20	2	0	1	2

Table 5.1 – Network simulation configurations for Test 1.1

5.2.1.1.1 - Results

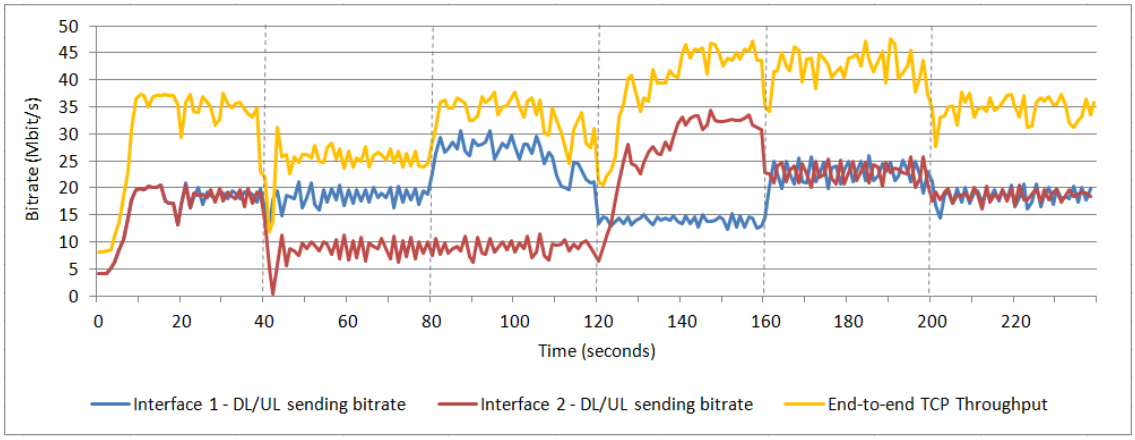

Figure 5.2 - Load-balancing for links with different bandwidths (Test 1.1)

In Figure 5.2, it can be observed that, as the bandwidth of each network interface changes from section to section, the load-balancing mechanism adapts the Packet Distribution so that each interface sends the DL/UL packets at a rate close to its available bandwidth, and consequently, the end-to-end TCP throughput from Server to Client is approximately the sum of the bandwidths of the two network interfaces, as can be seen in Figure 5.2. Since the packets of the end-to-end TCP flow are encapsulated in UDP packets, the end-to-end TCP throughput will always be lower than the sum of the sending bitrates of the two interfaces, due to the additional overhead caused by the headers that encapsulate the original packets. In any case, it can be concluded that the system is able to provide higher aggregated bandwidth, which was one of the main goals of this project.

5.2.1.2 - Test 1.2 - Performance with DOWNLOAD/UPLOAD and REAL-TIME traffic

In this test the bandwidth of both interfaces are limited to 25 Mbit/s throughout the 6 testing sections. The available bandwidth is only influenced by a REAL-TIME flow that will have a different bitrate for each section, which will be transmitted along with the DL/UL traffic. This test aims to show how the system balances the DL/UL traffic depending on the available bandwidth conditions of each link caused by parallel traffic flows of other types of traffic.

The network configurations for this test can be seen in Table 5.2, and are kept unchanged for the 6 testing sections. The REAL-TIME flow consists of a UDP flow generated using Iperf, from the Server to the Client. The sending bitrate of this flow for each section can be seen in Table 5.3. The DL/UL traffic was generated using Iperf (TCP flow), as in the previous test (Test 1.1).

	Bandwidth (Mbit/s)	RTT (ms)	Packet Loss (%)	Fin. Cost	RT Cost
Link 1	25	2	0	2	1
Link 2	25	2	0	1	2

Table 5.2 - Network simulation configurations for Test 1.2

	Section 1 (0s to 39s)	Section 2 (40s to 79s)	Section 3 (80s to 119s)	Section 4 (120s to 159s)	Section 5 (160s to 199s)	Section 6 (200s to 249s)
RT Bitrate (Mbit/s)	0	10	20	5	15	0

Table 5.3 – End-to-end sending bitrate of REAL-TIME flow for Test 1.2

5.2.1.2.1 - Results

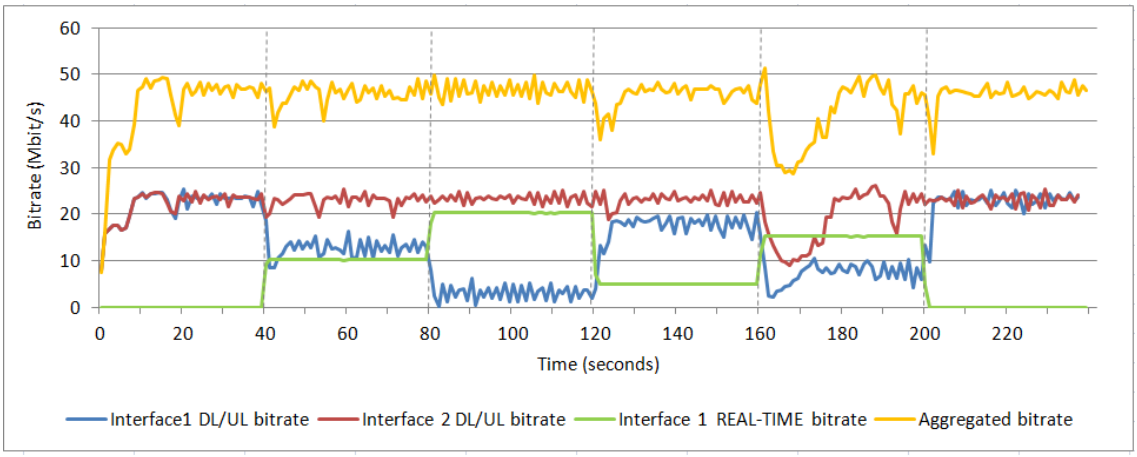

Figure 5.3 - Load-balancing with different types of traffic (Test 1.2)

In this test it can be observed (in Figure 5.3) that, as the bitrate of the REAL-TIME traffic changes from section to section, the load-balancing mechanism adjusts the Packet Distribution of the DL/UL traffic, so that the load on each network interface is adjusted to the available bandwidth of each interface. In this case, as expected, the REAL-TIME traffic is sent only through Interface 1, since it is the one with the lowest REAL-TIME cost, and there is no packet loss in the corresponding link. Hence, only the bitrate of the DL/UL sent by Interface 1 is affected by the REAL-TIME traffic. It can also be observed that, as expected, the aggregated bitrate of all the traffic is approximately the sum of the bandwidths of the two interfaces (50 Mbit/s), except at the beginning of section 5 (160s), where there is a significant drop in the aggregated bitrate. This is due to the sudden increase of 10 Mbit/s in

the REAL-TIME traffic bitrate, which momentarily congested Link 1, leading to a drop in the end-to-end TCP bitrate, while the system adjusts the Packet Distribution of the DL/UL traffic to the new link conditions.

5.2.2 - Links with different delays

These tests aim to evaluate the performance of the system when links with different delays are used to transfer the network traffic. As explained in Section 2.4.2, splitting traffic flows to transfer them over multiple paths with different delays causes the packets to arrive at the destination with a different order with respect to what the source has sent.

In Test 2.1, it is evaluated the ability of the system to reorder the data packets at the receiver, leading to a higher TCP throughput when compared with multipath transferring without reordering. In Test 2.2, the performance of the INTERACTIVE packet scheduling mechanism is evaluated, as the delays of the available links change.

5.2.2.1 - Test 2.1 - Performance with DOWNLOAD/UPLOAD traffic

In this test the two links were configured with the same constant bandwidth (25 Mbit/s) but with variable delays that change from section to section. The delay of Link 1 remains constant at 5 milliseconds and the delay of Link 2 is increased from section to section to values that can be seen in Table 5.4, which leads to an increasing difference between the delays of each link. Each testing section corresponds to a time interval of 20 seconds.

A TCP flow was generated with Iperf, from Server to Client, in order to evaluate the performance of the system with and without reordering the packets at the receiver module (the Client Module, in this case). The results for this test can be seen in Figure 5.4.

Another test was also done, with the same configurations, but this time an UDP flow was used. This flow had an end-to-end sending bitrate of 40 Mbit/s, generated using Iperf, to measure the packet loss caused by packets received out of order at the receiving application (Iperf client). Please notice that, as mentioned in Section 4.7, the current prototype is configured to identify UDP packets as REAL-TIME traffic, but that configuration was changed for this test in order to use UDP traffic as DL/UL traffic. The results for this test can be seen in Table 5.5.

		Bandwidth (Mbit/s)	Delay (ms)	Packet Loss (%)	Fin. Cost	RT Cost
Section 1 (0s to 19s)	Link 1	25	5	0	2	1
	Link 2	25	5	0	1	2
Section 2 (20s to 39s)	Link 1	25	5	0	2	1
	Link 2	25	10	0	1	2
Section 3 (40s to 59s)	Link 1	25	5	0	2	1
	Link 2	25	25	0	1	2
Section 4 (60s to 79s)	Link 1	25	5	0	2	1
	Link 2	25	50	0	1	2
Section 5 (80s to 99s)	Link 1	25	5	0	2	1
	Link 2	25	100	0	1	2
Section 6 (100s to 119s)	Link 1	25	5	0	2	1
	Link 2	25	150	0	1	2

Table 5.4 - Network simulation configurations for Test 2.1

5.2.2.1.1 - Results

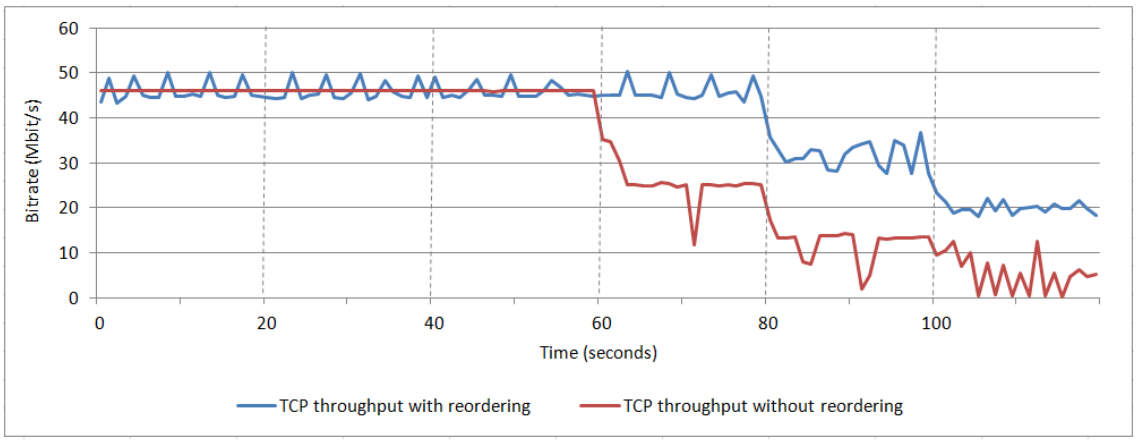

Figure 5.4 - End-to-end TCP throughput with and without packet reordering

		Average Pacekt Loss (%)	
	Delay difference (ms)	With Reordering	Without Reordering
Section 1 (0s to 19s)	0	0.32	27.79
Section 2 (20s to 39s)	5	1.08	37.52
Section 3 (40s to 59s)	20	4.31	42.51
Section 4 (60s to 79s)	45	12.78	25.08
Section 5 (80s to 99s)	95	13.87	25.11
Section 6 (100s to 119s)	145	32.52	42.52

Table 5.5 – Packet Loss with and without packet reordering

It was noticed that, after section 4 the TCP throughput (that can be seen in Figure ???) of the reordered traffic is significantly higher than the not reordered traffic. The not reordered traffic only starts to significantly decrease at section 4, where the difference between the delays of the two links is 45 ms. This is due to the fact that the TCP protocol, by itself, also provides packet reordering, which can attenuate, to some extent, the effects of the difference in the delays of the two links.

It was also observed that the measured end-to-end average packet loss, shown in Table 5.5, which in this case is caused by out of order packets, is much lower when the mechanism for reordering the packets at the receiver module is being used, even at lower delay differences.

It can be concluded that the packet reordering mechanism is performing well and that packet reordering at the receiver module is important to provide higher throughput to traffic flows, especially when the difference between the delays of the links used for multipath transferring the traffic from one module to the other is relatively high, and if the end-to-end applications, or used transport protocols, do not provide packet reordering suitable for such conditions.

It was also observed that in section 6 the end-to-end TCP throughput is lower than the bandwidth of one single link. Hence, it can be concluded that for higher delay differences it is preferable to send the TCP flows over only one path, instead of spreading it over multiple paths with high delay differences. An alternative to this would be to dynamically adjust the holding times of the buffers used for reordering based on the highest delay of the two links. These two approaches should be looked at in future work in order to find the best solution to this issue and, as a result, improve the performance of the system.

5.2.2.2 - Test 2.2 - Performance with INTERACTIVE traffic

In order to simulate, in a simplistic way, the INTERACTIVE traffic generated by bus users when browsing the web, it was used a web browser at the Client with several tabs randomly

performing HTTP requests to different web pages. The Server, in turn, was configured to forward this traffic, received from the Client, to an Internet gateway, and forward back to the Client the traffic that comes from the web servers as a response to those HTTP requests.

To test the INTERACTIVE packet scheduling mechanism, it was defined 7 testing sections, with 20 seconds time intervals, where the links have different delays. The configurations for each defined section can be seen in Table 5.6.

		Bandwidth (Mbit/s)	RTT (ms)	Packet Loss (%)	Fin. Cost	RT Cost
Section 1 (0s to 19s)	**Link 1**	25	10	0	1	1
	Link 2	25	10	0	2	2
Section 2 (20s to 39s)	**Link 1**	25	10	0	1	1
	Link 2	25	20	0	2	2
Section 3 (40s to 59s)	**Link 1**	25	40	0	1	1
	Link 2	25	20	0	2	2
Section 4 (60s to 79s)	**Link 1**	25	10	0	1	1
	Link 2	25	100	0	2	2
Section 5 (80s to 99s)	**Link 1**	25	100	0	1	1
	Link 2	25	100	0	2	2
Section 6 (100s to 119s)	**Link 1**	25	50	0	1	1
	Link 2	25	20	0	2	2
Section 7 (120s to 139s)	**Link 1**	25	22	0	1	1
	Link 2	25	20	0	2	2

Table 5.6 - Network simulation configurations for Test 2.2

5.2.2.2.1 - Results

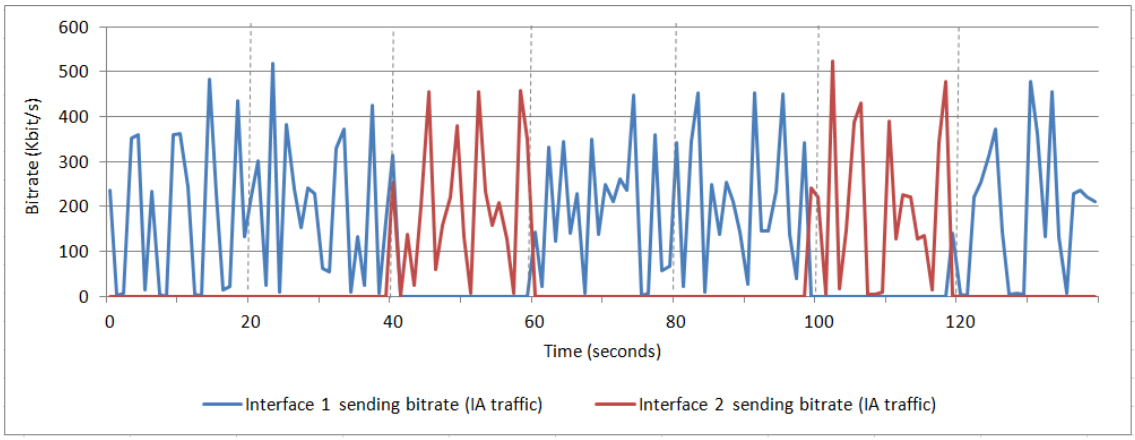

Figure 5.5 - INTERACTIVE traffic distribution (Test 2.2)

In Figure 5.5, it can be seen that, as expected, for each section, the system uses the link with the lowest RTT, unless the difference between the RTT of the two links is lower than 5 ms, as explained in 4.8.4. This particular behavior can be observed in section 7 of this test, where the RTT of link 1 is 22 ms and the RTT of link 2 is 20 ms. Although the RTT of link 1 is higher than the RTT of link 2 in that section, the difference between them is only 2 ms, and therefore the traffic is sent by Interface 1, which is the one with the lowest Financial Cost.

5.2.3 - Links with different packet loss

5.2.3.1 - Test 3 – Performance with REAL-TIME traffic

This test aims to test the system performance when using links with different packet losses. In order to achieve that, it was defined 6 testing sections where each link is configured to have a defined constant packet loss ratio. These configurations are shown in Table 5.7.

To test the performance of the REAL-TIME packet scheduling mechanism when using links with different packet loss, it was used a UDP flow generated using Iperf, which allows us to study the packet loss ratio experienced by the end-to-end traffic as the conditions of the two links change.

		Bandwidth (Mbit/s)	RTT (ms)	Packet Loss (%)	Fin. Cost	RT Cost
Section 1 (0s to 39s)	Link 1	25	10	0	1	1
	Link 2	25	10	0	2	2
Section 2 (40s to 79s)	Link 1	25	10	5	1	1
	Link 2	25	10	0	2	2
Section 3 (80s to 119s)	Link 1	25	10	0	1	1
	Link 2	25	10	0	2	2
Section 4 (120s to 159s)	Link 1	25	10	10	1	1
	Link 2	25	10	5	2	2
Section 5 (160s to 199s)	Link 1	25	10	5	1	1
	Link 2	25	10	15	2	2
Section 6 (200s to 239s)	Link 1	25	10	0	1	1
	Link 2	25	10	0	2	2

Table 5.7 - Network simulation configurations for Test 3

5.2.3.1.1 - Results

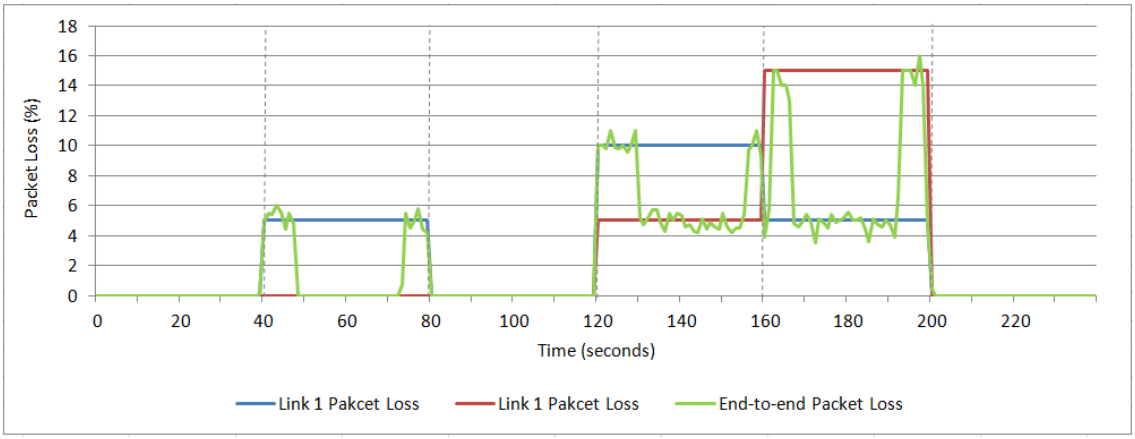

Figure 5.6 - End-to-end packet loss (Test 3)

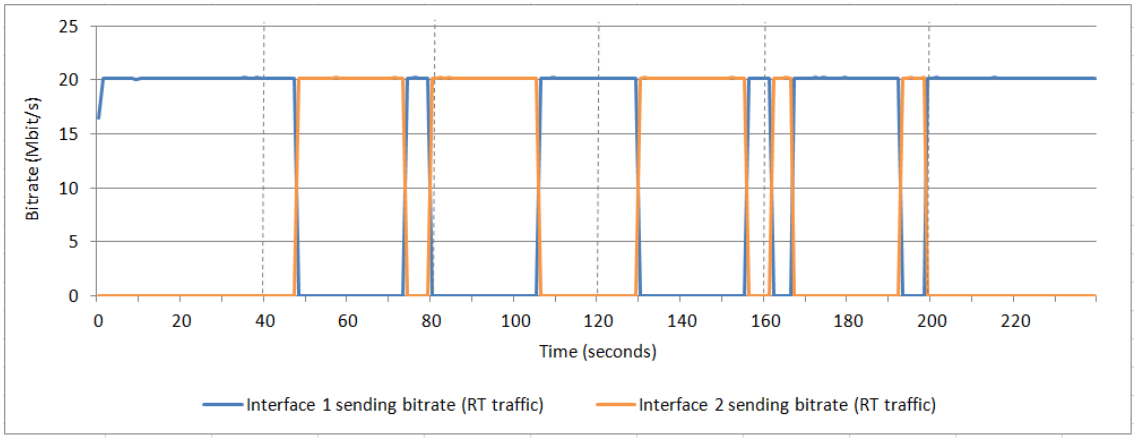

Figure 5.7 - REAL-TIME traffic distribution (Test 3)

	Packet Loss (%)		
	Link 1	Link 2	End-to-end (average)
Section 1	0	0	0
Section 2	5	0	1.81
Section 3	0	0	0
Section 4	10	5	6.69
Section 5	5	15	7.53
Section 6	0	0	0

Table 5.8 – Packet Loss for each testing section (Test 3)

As explained in Section 4.8.3, in ideal conditions (no packet loss in both links) the system sends the REAL-TIME traffic through the interface that corresponds to the link with the least REAL-TIME cost (Interface 1), as it can be seen in Figure 5.7. When there is packet loss above the maximum value (1%) on that link, after 5 seconds of consistent packet loss, the system declares Link 1 a Low Quality Link, and starts sending through Interface 2. After a defined time (20 seconds) the system checks again the quality of Link 1 by sending the traffic through Interface 1, which causes some packet loss, since Link 1 still has a packet loss ratio of 5%. This can be observed in section 2 of this test, in Figure 5.6 and Figure 5.7.

It can also be observed, for example, in section 7 of the above mentioned figures, that, as expected, when the two links are Low Quality Links, the system sends the RT traffic over the one with the lowest Packet Loss Ratio.

It can be concluded that the REAL-TIME packet scheduling mechanism is functioning properly and that its algorithm provides much lower packet loss to the end-to-end traffic, than if the system were to simply use the link with the least REAL-TIME cost, without taking into consideration the quality of the link in terms of packet loss. This can be seen in Table 5.8 , where, although the average end-to-end packet loss is always slightly higher than the packet loss of the link with the lowest packet loss, it is much lower than the link with the highest packet loss.

The packet loss experienced by the end-to-end traffic could be even lower if the waiting time that a link stays declared as Low Quality Link was higher (the current waiting time is 20 seconds). This waiting time could be tweaked in the future, to higher or lower values, to better address the conditions and requirements of the real life networks. An alternative to this would be to implement a mechanism that keeps sending probing traffic through the Low Quality Link to estimate its packet loss while the user generated traffic is being sent by another link. This way, user generated traffic wouldn’t be wasted as much, since that link would remain a Low Quality Link as long as it is having packet loss consistently.

5.2.4 - Conclusions

The performed isolated tests provided a better view of the operation of the several control and data transferring mechanisms that compose the Multipath Transfer Solution. They also allowed us to validate the good operation of such mechanisms, as well as finding some of their limitations that could be improved in the future, such as the issue described in Section 5.2.2.1, where we noticed that, for links with very different delays, the aggregated end-to-end TCP throughput is lower than the available bandwidth of just one link, which makes it preferable to send the traffic over a single path when under those conditions, or as an alternative, have the system dynamically adjust the holding times of the buffers used for reordering, based on the highest delay of the two links.

5.3 - Overall System Performance Tests

Section 5.2 presented a series of tests that were performed to individually evaluate the operation of the several mechanisms that compose the developed solution. This section presents an overall system performance test that aims to evaluate the performance of the developed solution as a whole, when operating under dynamically changing network conditions, and, at the same time, compare its performance with the performance of a system with SITMe-like behavior, under the same conditions. In order to do this, it was designed a simulation scenario that aims to represent, in a simplistic way, some of the real life network resources that could be available to a SITMe bus as it moves through the city.

5.3.1 - Simulation scenario

The simulation scenario designed for these tests consists of 8 defined zones, where the bus moves from Zone 1 to Zone 8 at a pace of 60 seconds per zone. Each zone has two available networks (UMTS and Wi-Fi) with different network conditions, such as the bandwidth, RTT and packet loss, for each zone.

In these tests the Interface 1 represents the UMTS interface, and therefore, always connects to the UMTS network. The same applies to the Interface 2 that, in turn, represents the Wi-Fi network interface.

In order to simulate Wi-Fi network migration as the bus moves through the city, there are 4 different Wi-Fi networks with different IP addresses. Every time the bus (Client Module) enters a zone with a different network, there is a time span of about 5 seconds where the corresponding network interface is offline. The UMTS network is available throughout the whole course, except for the beginning of Zone 8, where it is offline during 10 seconds, to simulate loss of connection, caused by, for instance, a section of the course where the power of the signal is too low.

The IP addresses of the Server's interfaces (192.168.10.2/24 for Interface 1 and 192.168.20.2/24 for Interface 2) remain unchanged, since in a real life situation the Server would have two public static IP addresses. Although it was mentioned that the Client connects to different Wi-Fi networks, in order to avoid using a router to connect the Client to the Server, for reasons already explained in Section 5.1, in this simulation scenario, all the IP addresses corresponding to the different Wi-Fi networks in the simulation, belong to the same network (192.168.20.0/24), which is also the same as the IP address of the corresponding Server's network interface. The same applies to the address of the UMTS interface, which belongs to the same network (192.168.10.0/24) as its corresponding Server interface.

The network properties of each testing zone can be seen in Table 5.9. Please notice that those network properties are from the Client's perspective, i.e., they represent the network resources available to the bus, seen from its perspective. In order to change these conditions from section to section, two Bash scripts were developed (one that runs at the Client and the other at the Server), that automatically run the commands of the network simulation tools, as well as the *ifconfig* command, used to change the IP address of the network interfaces and switching them on and off, so that the network conditions of each link match the designed simulation scenario.

The described simulation scenario is illustrated in Figure 5.8.

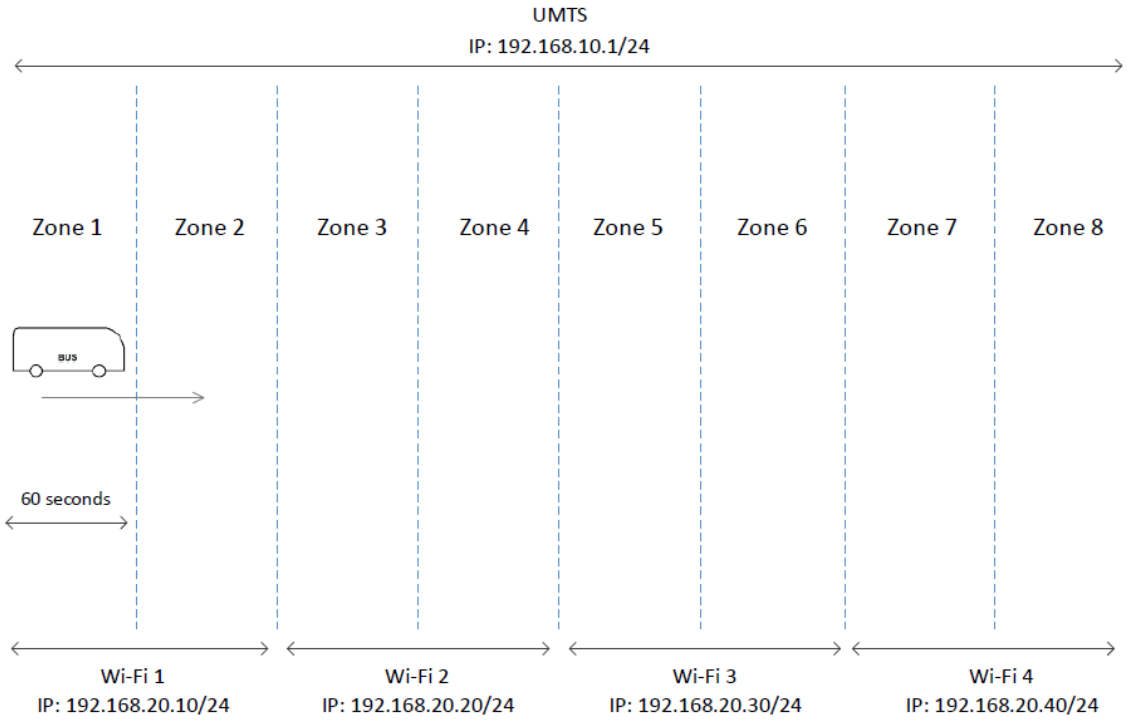

Figure 5.8 – Simulation scenario (Test 4.1 and 4.2)

			Bandwidth (Mbit/s)			Packet Loss (%)			
		IP Address	Uplink	Downlink	RTT (ms)	Uplink	Downlink	Fin. Cost	RT Cost
Zone 1 (0s to 59s)	**UMTS**	192.168.10.1/24	3	5	50	0	0	2	1
	Wi-Fi	192.168.20.10/24	5	10	6	0	0	1	2
Zone 2 (60s to 119s)	**UMTS**	192.168.10.1/24	5	10	60	0	0	2	1
	Wi-Fi	192.168.20.10/24	5	10	10	0	0	1	2
Zone 3 (120s to 179s)	**UMTS**	192.168.10.1/24	5	15	64	1	1	2	1
	Wi-Fi	192.168.20.20/24	10	25	8	0	0	1	2
Zone 4 (180s to 239s)	**UMTS**	192.168.10.1/24	10	25	46	0	0	2	1
	Wi-Fi	192.168.20.20/24	6	18	8	8	8	1	2
Zone 5 (240s to 299s)	**UMTS**	192.168.10.1/24	12	12	40	5	0	2	1
	Wi-Fi	192.168.20.30/24	5	30	16	2	0	1	2
Zone 6 (300s to 359s)	**UMTS**	192.168.10.1/24	8	20	80	0	0	2	1
	Wi-Fi	192.168.20.30/24	2	8	30	0	0	1	2
Zone 7 (360s to 419s)	**UMTS**	192.168.10.1/24	3	6	50	10	10	2	1
	Wi-Fi	192.168.20.40/24	5	25	50	3	3	1	2
Zone 8 (420s to 479s)	**UMTS**	192.168.10.1/24	15	15	60	0	0	2	1
	Wi-Fi	192.168.20.40/24	10	15	6	0	0	1	2

Table 5.9 – Configurations for each zone of the simulation scenario (Test 4.1 and 4.2)

5.3.2 - Test 4.1 - Performance of the Multipath Transfer Solution

This test aims to evaluate the performance of the Multipath Transfer Solution, as a whole, when transmitting several types of traffic using the available network resources, with properties that are constantly changing. To simulate those dynamic conditions, and observe how the system adapts to such conditions, in this test, the simulation scenario described in Section 5.3.1was adopted, where the bus goes from Zone 1 to Zone 8, spending 60 seconds at each zone.

In order to simulate, in a simplistic way, the network traffic generated by the bus passengers, three different traffic flows were generated (using Iperf) in both directions (from Client to Server and vice versa). To simulate the REAL-TIME and the DOWNLOAD/UPLOAD traffic, an UDP and a TCP flows were generated, respectively. The UDP flow has a constant sending bitrate of 1 Mbit/s. To simulate the burstiness of the INTERACTIVE traffic, a Bash script was used to generate UDP traffic (using Iperf) with random bandwidths (from 100

Kbit/s to 1 Mbit/s), during a random time span (from 1 to 5 seconds), separated by another random time span (from 1 to 3 seconds).

5.3.2.1 - Results

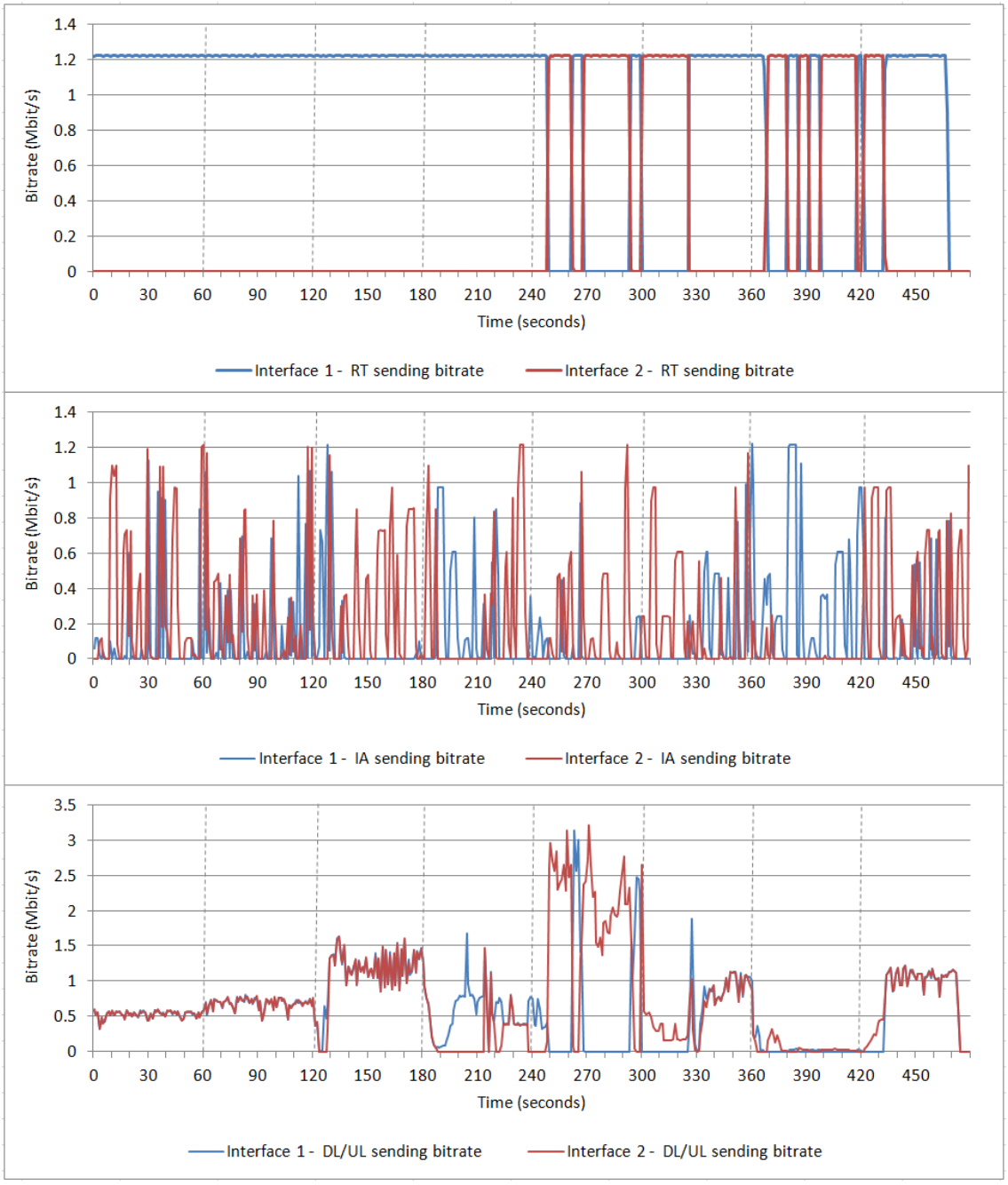

Figure 5.9 – Client Module - Sending bitrates of each type of traffic (Test 4.1)

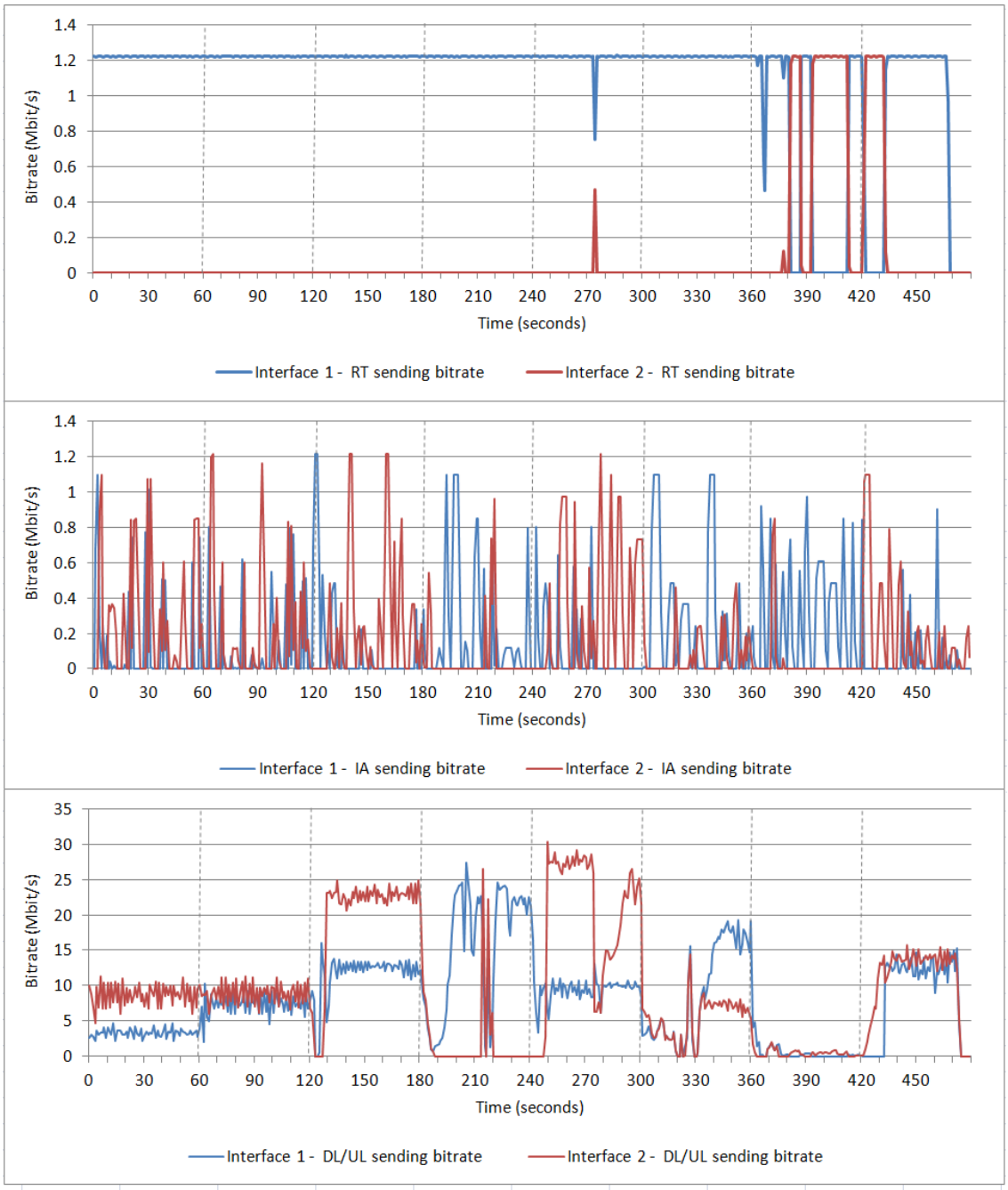

Figure 5.10 - Server Module - Sending bitrates of each type of traffic (Test 4.1)

In Figure 5.9 and Figure 5.10 it can be observed that, throughout the test, the system behaves as expected, by mainly using the interfaces with the lowest packet loss, when there is at least one Low Quality Link. This can be seen, for example in Zone 5, where the Client is mainly using Interface 2, which is the one with the lowest packet loss (2%), and at the same time, the Server, since there is no packet loss in none of the links in the Server to Client direction, uses Interface 1, which is the one with the lowest RT Cost, for sending RT traffic, and uses Interface 2 to send the INTERACTIVE traffic, since it is the one with the lowest RTT.

It can also be observed that when both links are Low Quality Links, as it happens in Zone 7, the bitrate of the DL/UL is seriously degraded, which was somehow expected since the TCP

traffic is being sent over links with a relatively high packet loss. In this zone it can also be seen that the RT traffic is shifted from one interface to the other too often before it stabilizes in the interface with the lowest packet loss. This might be due to some minor bug on the RT packet scheduling mechanism, or some momentary problem that is preventing the packet loss estimation mechanism from accurately estimating the packet loss ratio of the links. Although this is not a major issue, it is something that should be looked at in future work.

Every time the bus enters a zone with a different network, it was noticed that the DL/UL sending bitrate decreases drastically. This happens because it takes 3 seconds for the system to identify an interface as offline, hence, during that time, it doesn't know that the interface is offline and therefore continues to send traffic over it, which seriously degrades the end-to-end TCP throughput, during that time span.

At the beginning of Zone 8, the UMTS network is unreachable for about 10 seconds, as it was planned. It can be seen that, as expected, during that time, the system starts sending all the traffic through Interface 2, since it is the only one online, and when Interface 1 becomes online again, the system resumes its normal packet distribution behavior over the two interfaces.

The fact that the system is able to change the IP of its network interfaces from zone to zone, and still be able to receive traffic from the Server after that, as well as stopping the Server from sending traffic to the interface when it is turned off, can lead to the conclusion that the HELLO System is performing as expected.

Despite some identified issues, it can be concluded that in general the system as a whole is performing well, with the REAL-TIME traffic being sent mainly over the link with the lowest RT Cost, as well as the INTERACTIVE traffic being sent mainly over the link with the lowest RTT, and the DL/UL traffic being balanced over the two available links according to their available bandwidths, which are influenced by the other traffic flows being transferred at the same time.

5.3.3 - Test 4.2 - Performance with SITMe-like behavior

This test aims to evaluate the performance of a system with SITMe-like behavior in terms of usage of the available network resources, and compare it with the performance of the Multipath Transfer Solution evaluated in Section 5.3.2.

Although the network architecture of the SITMe was designed to support intelligent link selection based on several metrics, in practice, due to budget and time limitations, the prototype implementation used in the real pilot only accounted for the Financial Cost metric in order to switch between preferred public links. Hence, for the sake of simplicity, in order to simulate the SITMe behavior in a simplistic way, in this test it was only considered the Financial Cost metric as a deciding factor, i.e., the prototype was configured to preferably send all the traffic through the network interface with the lowest Financial Cost, which in this case is the Wi-Fi interface (Interface 2), when it is online.

Since the main objective of this test is to compare the performance of a SITMe-like system with the Multipath Transfer Solution, in this test, the simulation scenario and the

methods used to generate the traffic were the same as the ones used in the previous test, described in Section 5.3.2.

5.3.3.1 - Results

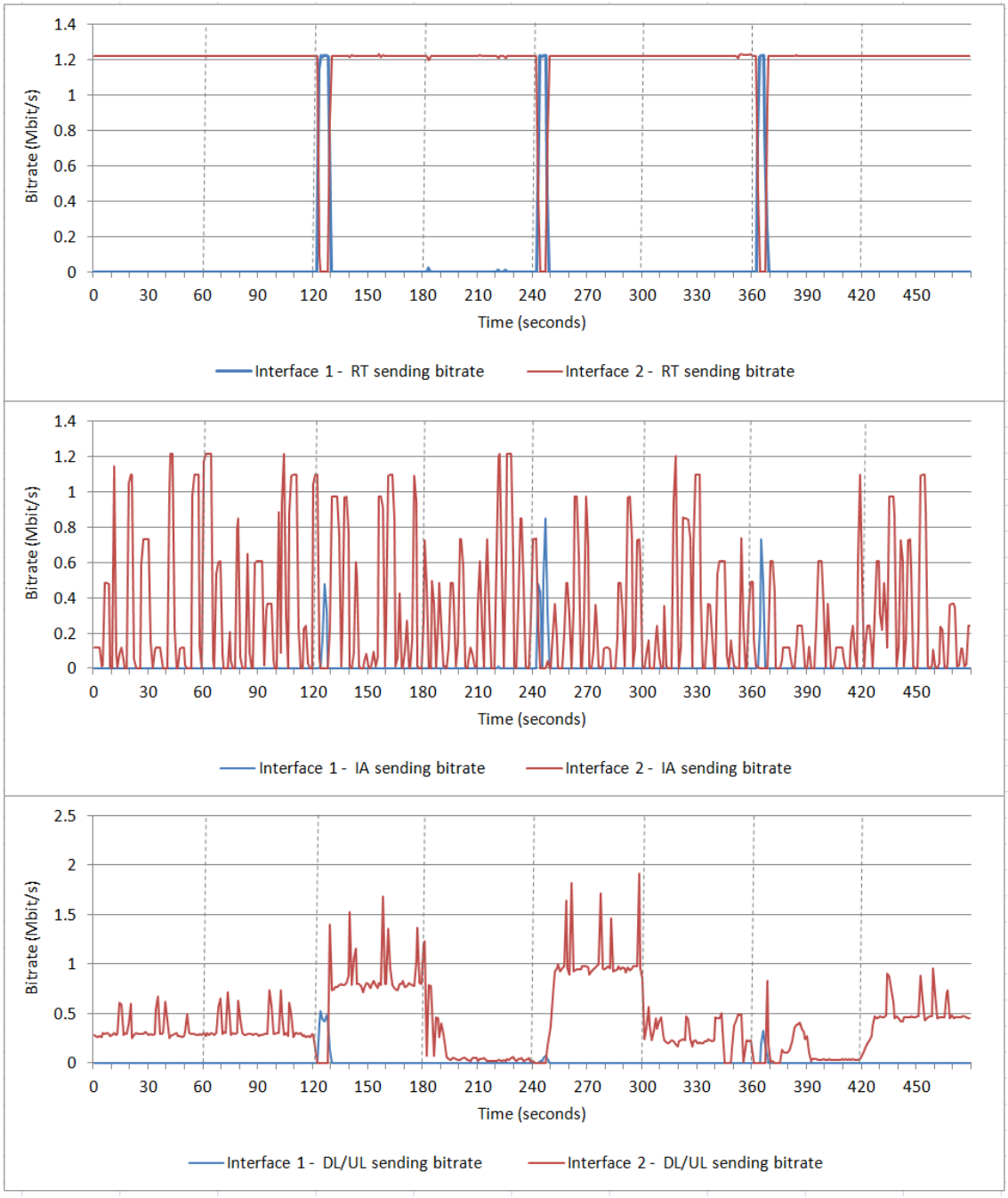

Figure 5.11 - Client Module - Sending bitrates of each type of traffic (Test 4.2)

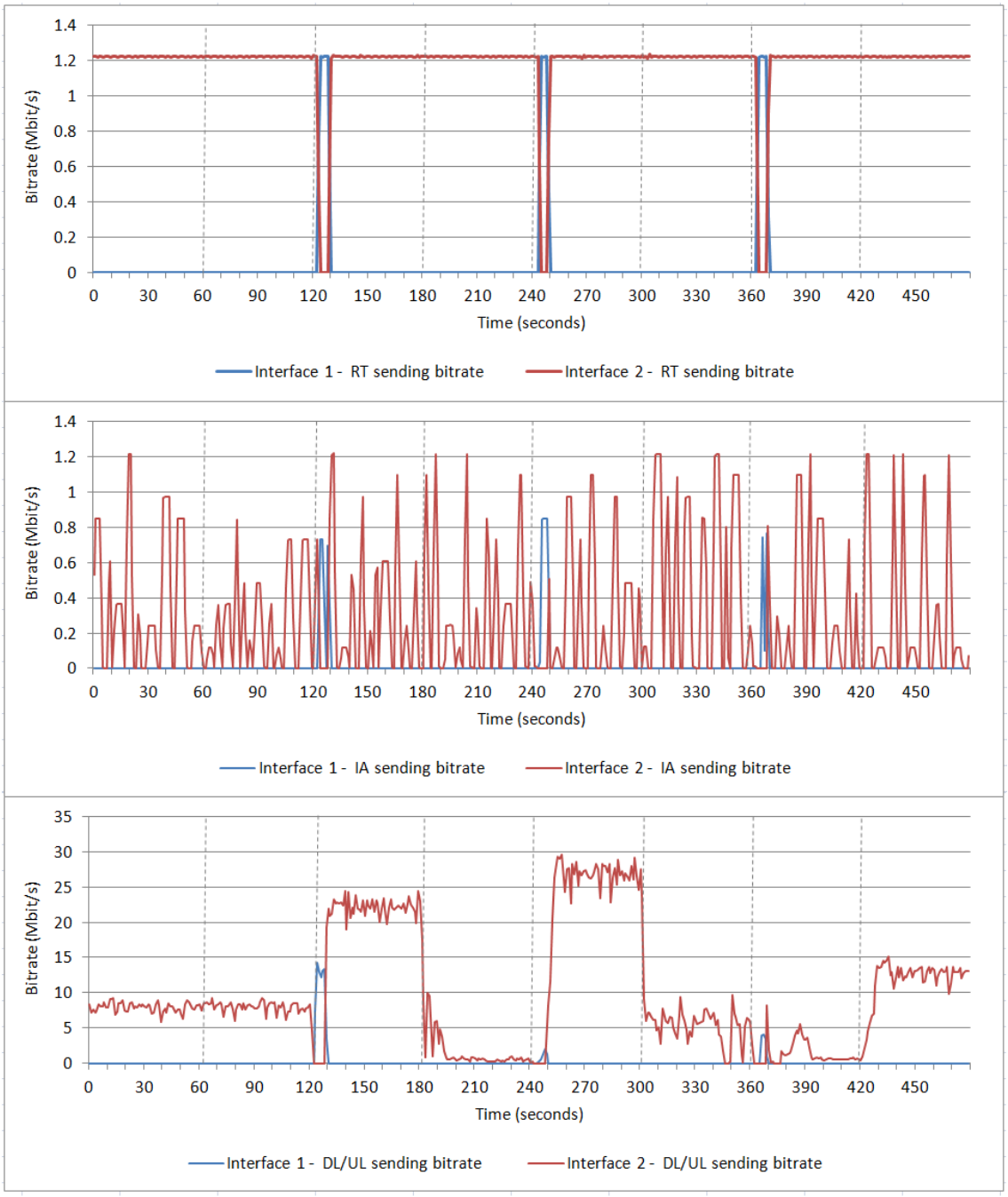

Figure 5.12 - Server Module - Sending bitrates of each type of traffic (Test 4.2)

It can be observed, in Figure 5.11 and Figure 5.12, that the system only uses the Interface 2 for the majority of the course, except when Interface 2 is offline due to network migration (in zones 3, 5 and 7). When compared to the Multipath Transfer Solution evaluated in Section 5.3.2, this is a big disadvantage, especially when the link with the lower Financial Cost (Wi-Fi), is also the one with the highest packet loss. This can be seen, for instance, in the DL/UL sending bitrate charts in Zone 4 (the Wi-Fi link has a packet loss of 8% and the UMTS link has 0%), where the sending bitrate of the SITMe-like system is much lower than the bitrate provided by the Multipath Transfer Solution for the same zone. This can be seen, by

comparing the DL/UL sending bitrate charts in Figure 5.10 and Figure 5.12, for the Multipath Transfer Solution and the SITMe-like system, respectively.

In addition to that, the SITMe-like system doesn't take advantage of the properties of each link, such as the RTT and REAL-TIME Cost, to transfer each type of traffic, neither does it provide multipath transfer with load-balancing to the DL/UL traffic, which leads to a much lower end-to-end throughput.

5.3.4 - Conclusions

The overall system performance tests provided a better view of the operation of the developed solution as a whole, as well as a comparison between the developed Multipath Transfer Solution and a system with SITMe-like behavior, by testing the two systems using the same network simulation scenario and traffic conditions. This simulation has, however, some limitations due to the simplicity of the simulation scenario and used methods, but, in any case, we feel it is good enough to validate, to some extent, the good operation of the solution as a whole and compare its performance with a SITMe-like system.

By comparing the performance of the two types of systems, in Section 5.3.2 and Section 5.3.3, it can be concluded that, despite some minor issues that should be looked at in the future, the Multipath Transfer Solution is an improvement to the SITMe project, since it improves the usage of the available network resources, by dynamically adapting its behavior to take advantage of the properties of the networks it uses to transfer the user generated traffic between each module (Client and Server) of the system. Unlike the current SITMe system, it provides features such as distribution of the different types of traffic according to the defined metrics, such as the RTT and REAL-TIME Cost, and also provides higher bandwidth aggregation, by balancing the traffic over the available links.

Chapter 6

Conclusions and Future Work

6.1 - Conclusions

The main goal of this dissertation was to develop a solution to a specific SITMe system's limitation, which is the underutilization of some of the available network resources. SITMe only uses the best available network link at each moment, according to some defined metrics, leaving other available links unused. In order to solve this problem, a Multipath Transfer Solution was developed to provide intelligent traffic distribution and load-balancing over the available network links (based on defined link metrics and type of traffic), which results in higher aggregated bandwidth and fault-tolerance to the SITMe project's communication system. Although this solution was developed to improve the SITMe system, it is also generic enough so that it can also be easily adapted to other heterogeneous network systems. In summary, this solution provides a multipath abstraction layer for multihomed devices and routers, allowing to intelligently explore the multiple paths available for the Internet as if they were only one link. This introduces functionalities such as multipath bandwidth aggregation for standard TCP and UDP flows initiated between legacy clients and servers, and seamless network mobility for legacy terminals.

There were some implementation challenges, such as the implementation of the available bandwidth estimation mechanism, that lacked the accuracy and speed necessary to be used by the load-balancing mechanism. This led us to use an alternative approach to assess the link congestion, based in the socket send queues used to send the traffic associated to each link. This alternative proved to work well, but, as explained in Section 4.8.1, it has some limitations that can make it useless in a real network scenario. In summary, this method of load-balancing based on the socket send queues only works well, at the moment, if the congestion is caused by the uplink speed of the local network Interface and not by a bottleneck in the traffic path to the destination node (e.g. Wi-Fi connection to a AP/Router with uplink bandwidth lower than the bandwidth of the Wi-Fi link) as it only obtains status feedback from the local queues.

A series of tests were performed, with favorable results, to individually evaluate the operation of the several mechanisms that constitute the Multipath Transfer Solution, as well as the overall performance of the system as a whole. It was also performed a test to the system, configured to act with a SITMe-like behavior, in order to compare its performance

with the Multipath Transfer Solution. This comparison, along with the results from all the other tests, allowed us to conclude that, despite some identified issues that should be looked at in the future, the Multipath Transfer Solution is performing well and has the potential to bring a substantial improvement to the SITMe project. This is achieved by both taking advantage of aggregated bandwidth and intelligently selecting the best paths for a given type of traffic, based on real-time evaluation of the links' characteristics.

6.2 - Known Limitations and Future Work

Although the main goals for this dissertation were achieved, there are still several improvements that can be made to the system in order to optimize its performance, QoE and cost of operation.

The current solution uses the REAL-TIME Cost metric as the main decision factor to choose the best link to be used for REAL-TIME traffic. This metric is mainly based on the availability of the network link, since the users of real-time applications expect a stable connection with the least interruptions as possible. In the current version of the prototype, this value is passed to the program at the start and remains static. However, the predicted availability might not stand true at all times, hence a mechanism to log the availability of each network interface and calculate this metric dynamically should be implemented.

There are currently 3 types of traffic established that are processed by the system in a different way according to their properties. However, the mechanism used to identify them is still very primitive. Hence, a new mechanism should be developed, as well as establishing additional types of traffic, in order to provide a better identification of the different types of traffic generated by the bus users.

There are some issues that should be addressed regarding the security of the system. The current solution offers little to no security. The data packets and the control messages are transferred unencrypted, and there is no data integrity or mutual authentication between the Client and the Server modules. This allows an attacker to impersonate a Client Module and register itself in the Server using messages of the HELLO System's protocol. This opens a gate to a series of possible attacks to the server, such as overloading the server with traffic generated by the attacker, or tricking the Server into forwarding to him the traffic belonging to another legitimate Client. An attacker can also replicate the control messages used by the link metrics estimation mechanisms to induce confusion and seriously degrade the system's performance, by sending those messages to the Client and Server with false values, which may lead to misestimation of the link metrics values. Therefore, these issues should be taken into consideration in the future in order to improve the security of the system, which is crucial in a real network scenario.

The developed solution uses public networks to transfer the network traffic. Many of those networks are behind a NAT[6], which means that the information about the Client Module's UDP ports to be used for data transferring from Server to Client, that is carried by the INTERFACE_REGISTRATION messages, becomes useless, since those ports would be

[6] Network Address Translation

unreachable. One way to solve this problem would be to use the same UDP port for signaling and for data transferring. This way, a UDP binding is established in the NAT by the INTERFACE_REGISTRATION messages sent periodically to the Server, which can be used for data transferring from Server to Client. For that to work, it would be necessary to implement a way to distinguish if the received packets are control messages or data packets, which is not currently implemented.

Another known limitation of this solution is that it relies on static gateways for the Server Module IP addresses in order to guarantee that the packets are sent by the intended network interface. Hence, every time a network interface of a Client Module connects to a public network, it has to configure the network gateway to the corresponding IP address of the Server Module. The gateway address to be configured is the same as the gateway address to the network the Client interface is connected to. This could be achieved by implementing on the Client Module a mechanism that captures the DHCP messages and retrieves the IP address of the gateway, and then configures the static gateway for the corresponding Server IP address.

As mentioned in Section 4.1, it is assumed that all the resources (network links) available to the system will be used simultaneously, even if one link has enough available bandwidth to satisfy the demand. In order to minimize costs of operation, it would be useful to implement a different approach that only uses a single link (the one with the lowest financial cost) until it is almost congested, and then starts using both if the demand is too high for that link.

As explained in Section 3.3.2, the metric used to determine the best link, in terms of latency, is the Round-Trip Time. This can be a limitation if the uplink and downlink delays of a certain network link are substantially asymmetric. For example, if, from the Client Module perspective, link 1 has delays of 5 ms (uplink) and 50 ms (downlink) and link 2 has delays of 50 ms (uplink) and 10 ms (downlink), the system would currently chose link 1 to send packets belonging to delay sensitive traffic, in both directions (from Client Module to Server Module and vice versa). This means that a packet would take a total of 55 ms (5+50) to go from client to server and from the server back to the client. However, if it was implemented a mechanism that estimates the one way delay of a link and use it as a metric, instead of the RTT, the overall delay experienced by the same packet would be of only 15 ms (5+10), since the system would use link 1 for uplink and link 2 for downlink.

In order to estimate the one way delay of each link, it would be necessary to have the clocks of the Client and Server Modules synchronized with at least 1 millisecond accuracy. However, synchronization with this kind of precision is not as easy to achieve as it might seem. Using a networking protocol for clock synchronization such as the Network Time Protocol (NTP) [27] would be the easier way to implement a solution for this issue, but, although NTP is able to achieve better than 1 millisecond accuracy in ideal conditions, e.g., in a Local Area Network (LAN), the errors can go up to 100 ms or more, over the Internet, especially if the uplink and downlink delays are asymmetric [28] , which is the scenario where the one way delay estimation would be most useful. The best approach to solve this problem would be to have the Client and Server Modules obtain the time from the same source, where the delay experienced by the signal provided by that source used to synchronize the clocks of both modules would be practically the same, such as a satellite signal. This could be achieved by having a GPS device in each module that can be used to synchronize the clocks, which would allow the system to estimate the one way delay metric with the required accuracy.

This solution currently uses a per-packet approach for performing the load-balancing of the traffic. However, in a scenario where there are several bus users connected to a Client Module generating several traffic flows, perhaps it would be better to use an approach based on the balancing of the traffic flows, i.e., sending each flow over a single path, which would minimize the effects experienced by traffic flows when transferred over a multipath transferring scheme, already mentioned in Section 4.6. The ideal approach would be to use the two methods (per-packet and per-flow balancing) in a way that the system could benefit from the advantages of both methods, by dynamically switching between the two based on the demand and network conditions.

As explained in Section 4.8.1, the available bandwidth estimation mechanism is currently not able to provide bandwidth estimations with the accuracy and speed required to be used as a metric for the load-balancing mechanism. The alternative based on the socket send queues, in spite of working well, has some limitations that can make it useless in a real life network scenario, as explained in Section 4.8.1. Hence, a new way to provide feedback to the load-balancing mechanism needs to be implemented. The ideal solution would be to develop a new available bandwidth estimation mechanism that is able to provide estimations that are accurate and frequent enough to be used as a metric by the load-balancing mechanism, without adding too much load to the network links.

References

1. Fontes, H., *Multi-Technology Router for Mobile Networks: Layer 2 Overlay Network over Private and Public Wireless Links*. 2010, Faculdade de Engenharia da Universidade do Porto

2. Oliveira, L., *Vehicular Networks: IEEE 802.11p Analysis and Integration into an Heterogeneous WMN*. 2012, Faculdade de Engenharia da Universidade do Porto.

3. Zimmermann, H., *OSI reference model – The ISO model of architecture for open systems interconnection*. 1980.

4. Ford, A., et al. *TCP Extensions for Multipath Operation with Multiple Addresses. Request For Comments 6824, IETF*. 2013 ; Last access: December, 2013; Available from: http://tools.ietf.org/html/rfc6824.

5. Postel, J. *Transmission Contro Protocol. Request For Comments 793, IETF*. 1981 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/rfc793.

6. Postel, J. *User Datagram Protocol. Request For Comments 768, IETF*. 1980 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/rfc768.

7. Lee, B.P., et al., *Avoiding congestion collapse on the Internet using TCP tunnels*. 2001.

8. Hondaa, O., et al., *Understanding TCP Over TCP: Effects of TCP Tunneling on End-to-End Throughput and Latency*. 2005.

9. R. Stewart, E. *Stream Control Transmission Protocol , Request For Comments 4960, IETF*. 2007 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/rfc4960.

10. Postel, J. *Internet Protocol. Request For Comments 791, IETF*. 1981 ; Last access: December, 2013; Available from: http://tools.ietf.org/html/rfc791.

11. Iyengar, J.R., *End-to-End Concurrent Multipath Transfer Using Transport Layer Multihoming* 2006, Faculty of the University of Delaware.

12. Singh, V., et al. *Multipath RTP (MPRTP). Internet-draft, IETF*. 2013 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/draft-singh-avtcore-mprtp-07.

13. Schulzrinne, H., et al. *RTP: A Transport Protocol for Real-Time Applications. Request For Comments 3550, IETF*. 2003 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/rfc3550.

14. Kohler, E., M. Handley, and S. Floyd. *Datagram Congestion Control Protocol (DCCP). Request For Comments 4340, IETF*. 2006 ; Last access: January, 2014; Available from: http://tools.ietf.org/html/rfc4340.

15. Kohler, E. *Datagram Congestion Control Protocol Mobility and Multihoming. Internet-draft, IETF*. 2004 ; Last access: January, 2014; Available from: http://tools.ietf.org/search/draft-kohler-dccp-mobility-00.

16. Moskowitz, R., et al. *Host Identity Protocol. Request For Comments 5201, IETF*. 2008 ; Last access: February, 2014; Available from: http://tools.ietf.org/html/rfc5201.

17. Nikander, P., et al. *End-Host Mobility and Multihoming with the Host Identity Protocol. Request For Comments 5206, IETF*. 2008 ; Last access: February, 2014; Available from: http://tools.ietf.org/html/rfc5206.

18. Hu, N. and P. Steenkiste, *Evaluation and Characterization of Available Bandwidth Probing Techniques*. 2003.

19. Jain, M. and C. Dovrolis, *Pathload: A measurement tool for end-to-end available bandwidth*. 2002.

20. Sommers, J., P. Barford, and W. Willinger, *A Proposed Framework for Calibration of Available Bandwidth Estimation Tools*. 2005.

21. Strauss, J., D. Katabi, and F. Kaashoek, *A Measurement Study of Available Bandwidth Estimation Tools*. 2003.

22. Novo, N., et al., *Video Streaming Over Multi-Radio Access Networks: An Access Aggregation Approach*. 2012.

23. Yabandeh, M., S. Zarifzadeh, and N. Yazdani, *Improving performance of transport protocols in multipath transferring schemes*. Computer Communications, 2007.

24. *NetEm*. ; Last access: June, 2014; Available from: http://manpages.ubuntu.com/manpages/raring/en/man8/tc-netem.8.html.

25. *Token Bucket Filter (tc-tbf)*. ; Last access: June, 2014; Available from: http://linux.die.net/man/8/tc-tbf.

26. *Iperf*. ; Last access: June, 2014; Available from: http://iperf.fr/.

27. Mills, D., J. Martin, and J. Burbank. *Network Time Protocol Version 4: Protocol and Algorithms Specification. Request For Comments 5905*. 2010 ; Last access: June, 2014; Available from: http://www.ietf.org/rfc/rfc5905.txt.

28. Mills, D. *Executive Summary: Computer Network Time Synchronization*. 2012 ; Last access: June, 2014; Available from: http://www.eecis.udel.edu/~mills/exec.html.

Appendix A

List of Control Messages

HELLO System:

1 - INTERFACE_REGISTRATION

MPTS|1|RBridgeID|InterfaceID|InterfaceNumber|DataPort|ABEcontrolPort|ABEprobingPort|FinCost|RTcost|

2 - REGISTRATION_COMPLETED

MPTS|2|

3 - RC_ACK

MPTS|3|

Available Bandwidth Estimation :

4 - STARTING_PROBING

MPTS|4|

5 - STARTING_PROBING_ACK

MPTS|5|

6 – PROBING_DONE

MPTS|6|

7 – PROBING_DONE_ACK

MPTS|7|

Packet Loss Estimation:

8 – LOST_PACKETS

MPTS|8|number_of_lost_packets|first_packet_number|last_packet_number|list_of_lost_packets|

9 – NO_ LOST_ PACKETS

MPTS|9 |last_packet_number|

RTT Estimation:

10 – PING_REQUEST

MPTS|10|SequenceNumber|

11 – PING_RESPONSE

MPTS|11|SequenceNumber|